Table of Contents

Introduction ................................................................. 17
Architecture ............................................................ 18
Getting Started ............................................................ 20
  1 Installing Thinfinity VirtualUI ........................................... 20
  2 Simple UI Remoting ......................................................... 23
      Compiling and Testing a WinForms Application .................................................. 24
      Compiling and Testing a Delphi Application ......................................................... 26
      Compiling and Testing a C++ Application ......................................................... 27
      Registering the Application in Thinfinity VirtualUI Server .................................. 30
      Accessing the App from the Web ........................................................................ 31
      Application Execution behavior ....................................................................... 32
  3 Adapting the Application .................................................. 33
      Customizing the Web Page ............................................................................. 34
      Preparing the Web Page ............................................................................. 35
      Customizing Settings ............................................................................. 37
      Programming VirtualUI ........................................................................ 39
      Handling VirtualUI Events ........................................................................ 40
Advanced Programming with jsRO ........................................... 42
  1 The Development Lab ....................................................... 43
  2 Life cycle of the jsRO Objects ........................................... 44
  3 Creating jsRO objects ......................................................... 46
  4 Updating Properties .......................................................... 48
  5 OnPropertyChange(), OnSet() y OnGet() events .................. 49
  6 jsRO Remote Calls ............................................................ 52
      Remote Methods ............................................................................. 54
      Custom Events ............................................................................. 57
Scaling and Load Balancing .................................................... 59
  1 Scaling and Load Balancing Configurations ...................... 59
  2 Installing components ....................................................... 62
  3 Enabling Multiple RDS Accounts ......................................... 63
  4 Configuring Load Balancing ................................................ 65
Enhanced Browser and DPI Support ........................................ 68
  1 Model Inheritance .......................................................... 69
  2 Property Reference .......................................................... 70
  3 The Calculation Process ..................................................... 73
  4 Examples ............................................................................. 73
One-Time URL .............................................................. 74
  1 How it Works ........................................................................ 75
  2 Creating a One-Time URL .................................................. 76
  3 Single Sign-On Sequence Example ....................................... 77

© 2015, Cybele Software, Inc.
Configuration Reference

1 Development Server................................................................. 78
   General .................................................................................. 80
   Applications .......................................................................... 81
      Application Profile ............................................................ 82
         General .......................................................................... 85
         Credentials ..................................................................... 86
         Permissions .................................................................... 87
      Weblink Profile ................................................................. 88
         Permissions ..................................................................... 90
   Licenses .................................................................................. 91

2 Production Server .................................................................. 92
   General .................................................................................. 94
      Configure HTTP Error Responses ...................................... 96
      Managing the SSL Certificate ............................................. 98
         The Default Embedded Certificate ................................ 101
         A Self-signed Certificate ................................................ 102
         A CA Certificate ............................................................ 104
      HTTPS Security Settings ................................................... 106
   Gateways .............................................................................. 106
   RDS ...................................................................................... 108
   Applications .......................................................................... 111
      Application Profile ............................................................ 113
         General .......................................................................... 115
         Credentials ..................................................................... 116
         Permissions .................................................................... 117
      Weblink Profile ................................................................. 118
         Permissions ..................................................................... 120
   Licenses .................................................................................. 121

3 Gateway ................................................................................ 122

Symbol Reference .................................................................... 124

1 ActiveX Interfaces ................................................................. 124
   VirtualUI Runtime Library ................................................... 124
      VirtualUI::JSDataType Enumeration ................................. 126
      VirtualUI::Orientation Enumeration ................................. 126
      VirtualUI::MouseMoveGestureStyle Enumeration .............. 126
      VirtualUI::MouseMoveGestureAction Enumeration ............ 127
   IVirtualUI Interface .............................................................. 127
      IVirtualUI Methods .......................................................... 128
         DevMode Method .......................................................... 129
         IVirtualUI::DevMode Method ([out, retval] VARIANT_BOOL*) 129
         IVirtualUI::DevMode Method ([in] VARIANT_BOOL) .......... 129
         Enabled Method ............................................................ 130
         IVirtualUI::Enabled Method ([out, retval] VARIANT_BOOL*) 130
         IVirtualUI::Enabled Method ([in] VARIANT_BOOL) .......... 130
      StdDialogs Method ........................................................... 130
         IVirtualUI::StdDialogs Method ([out, retval] VARIANT_BOOL*) 131
         IVirtualUI::StdDialogs Method ([in] VARIANT_BOOL) .......... 131
         Start Method ................................................................. 131
         IVirtualUI::Stop Method ................................................ 132
         IVirtualUI::Active Method .............................................. 132

© 2015, Cybele Software, Inc.
IClientSettings::MouseMoveGestureAction Method ([in] enum MouseMoveGestureAction)

IClientSettings::CursorVisible Method ([out, retval] VARIANT_BOOL*)

IClientSettings::Enabled Method ([in] VARIANT_BOOL)

IClientSettings::Port Method ([out, retval] long*)

IClientSettings::StartBrowser Method ([in] VARIANT_BOOL)

IClientSettings::ShowVirtualKeyboard Method

IBrowserInfo Interface ................................................................. 136

IBrowserInfo Methods ................................................................ 136

IBrowserInfo::ViewHeight Method .............................................. 137

IBrowserInfo::ViewHeight Method ([out, retval] long*) ............... 138

IBrowserInfo::ViewWidth Method .............................................. 138

IBrowserInfo::ViewWidth Method ([out, retval] long*) ............... 138

IBrowserInfo::ScreenHeight Method ......................................... 139

IBrowserInfo::ScreenWidth Method ......................................... 139

IBrowserInfo::Username Method .............................................. 140

IBrowserInfo::IPAddress Method .............................................. 140

IBrowserInfo::UserAgent Method .............................................. 140

IBrowserInfo::ScreenResolution Method .................................. 140

IBrowserInfo::Orientation Method .......................................... 141

IBrowserInfo::UniqueBrowserId Method ................................... 141

IBrowserInfo::GetCookie Method .............................................. 141

IBrowserInfo::SetCookie Method .............................................. 141

IBrowserInfo::Location Method .............................................. 142

IDevServer Interface .................................................................. 142

IDevServer Methods .................................................................. 142

IDevServer::Enabled Method ([out, retval] VARIANT_BOOL*) ........ 143

IDevServer::Enabled Method ([in] VARIANT_BOOL) ..................... 143

IDevServer::Port Method ([out, retval] long*) ......................... 144

IDevServer::Port Method ([in] long) ......................................... 144

IDevServer::StartBrowser Method ([out, retval] VARIANT_BOOL*) 145

IDevServer::StartBrowser Method ([in] VARIANT_BOOL) ........... 145

IClientSettings Interface ....................................................... 145

IClientSettings Methods ....................................................... 146

IClientSettings::CursorVisible Method ([out, retval] VARIANT_BOOL*) 146

IClientSettings::CursorVisible Method ([in] VARIANT_BOOL) .... 147

IClientSettings::MouseMoveGestureAction Method ................. 147

IClientSettings::MouseMoveGestureAction Method ([out] enum MouseMoveGestureAction*) 147

IClientSettings::MouseMoveGestureAction Method ([in] enum MouseMoveGestureAction) ................. 147

© 2015, Cybele Software, Inc.
<table>
<thead>
<tr>
<th>Interface/Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MouseMoveGestureStyle Method</td>
<td>148</td>
</tr>
<tr>
<td>IClientSettings::MouseMoveGestureStyle Method ([out, retval] enum_MOUSEMOVEGESTURERESTYLE*)</td>
<td>148</td>
</tr>
<tr>
<td>IClientSettings::MouseMoveGestureStyle Method ([in] enum_MOUSEMOVEGESTURERESTYLE)</td>
<td>148</td>
</tr>
<tr>
<td>IJSValue Interface</td>
<td>149</td>
</tr>
<tr>
<td>IJSValue Methods</td>
<td>149</td>
</tr>
<tr>
<td>AsBool Method</td>
<td>149</td>
</tr>
<tr>
<td>IJSValue::AsBool Method ([out, retval] VARIANT_BOOL*)</td>
<td>150</td>
</tr>
<tr>
<td>IJSValue::AsBool Method ([in] VARIANT_BOOL)</td>
<td>150</td>
</tr>
<tr>
<td>AsFloat Method</td>
<td>150</td>
</tr>
<tr>
<td>IJSValue::AsFloat Method ([out, retval] float*)</td>
<td>151</td>
</tr>
<tr>
<td>IJSValue::AsFloat Method ([in] float)</td>
<td>151</td>
</tr>
<tr>
<td>AsInt Method</td>
<td>151</td>
</tr>
<tr>
<td>IJSValue::AsInt Method ([out, retval] long*)</td>
<td>151</td>
</tr>
<tr>
<td>IJSValue::AsInt Method ([in] long)</td>
<td>152</td>
</tr>
<tr>
<td>AsJSONObject Method</td>
<td>152</td>
</tr>
<tr>
<td>IJSValue::AsJSON Method ([out, retval] BSTR*)</td>
<td>152</td>
</tr>
<tr>
<td>IJSValue::AsJSON Method ([in] BSTR)</td>
<td>152</td>
</tr>
<tr>
<td>AsString Method</td>
<td>153</td>
</tr>
<tr>
<td>IJSValue::AsString Method ([out, retval] BSTR*)</td>
<td>153</td>
</tr>
<tr>
<td>IJSValue::AsString Method ([in] BSTR)</td>
<td>153</td>
</tr>
<tr>
<td>AsType Method</td>
<td>153</td>
</tr>
<tr>
<td>IJSValue::AsType Method ([out, retval] enum_IJSDataType*)</td>
<td>154</td>
</tr>
<tr>
<td>IJSValue::AsType Method ([in] enum_IJSDataType)</td>
<td>154</td>
</tr>
<tr>
<td>RawValue Method</td>
<td>154</td>
</tr>
<tr>
<td>IJSValue::RawValue Method ([out, retval] VARIANT_BOOL*)</td>
<td>155</td>
</tr>
<tr>
<td>IJSValue::RawValue Method ([in] VARIANT_BOOL)</td>
<td>155</td>
</tr>
<tr>
<td>IJSNamedValue Interface</td>
<td>155</td>
</tr>
<tr>
<td>IJSNamedValue Methods</td>
<td>156</td>
</tr>
<tr>
<td>Name Method</td>
<td>156</td>
</tr>
<tr>
<td>IJSNamedValue::Name Method ([out, retval] BSTR*)</td>
<td>156</td>
</tr>
<tr>
<td>IJSNamedValue::Name Method ([in] BSTR)</td>
<td>156</td>
</tr>
<tr>
<td>IJSObject Interface</td>
<td>157</td>
</tr>
<tr>
<td>IJSObject Methods</td>
<td>157</td>
</tr>
<tr>
<td>Id Method</td>
<td>158</td>
</tr>
<tr>
<td>IJSObject::Id Method ([out, retval] BSTR*)</td>
<td>158</td>
</tr>
<tr>
<td>IJSObject::Id Method ([in] BSTR)</td>
<td>158</td>
</tr>
<tr>
<td>IJSObject::Properties Method</td>
<td>159</td>
</tr>
<tr>
<td>IJSObject::Methods Method</td>
<td>159</td>
</tr>
<tr>
<td>IJSObject::Events Method</td>
<td>159</td>
</tr>
<tr>
<td>IJSObject::Objects Method</td>
<td>159</td>
</tr>
<tr>
<td>IJSObject::FireEvent Method</td>
<td>160</td>
</tr>
<tr>
<td>IJSObject::ApplyChanges Method</td>
<td>160</td>
</tr>
<tr>
<td>IJSObject::ApplyModel Method</td>
<td>160</td>
</tr>
<tr>
<td>IJSObjects Interface</td>
<td>161</td>
</tr>
<tr>
<td>IJSObjects Methods</td>
<td>161</td>
</tr>
<tr>
<td>IJSObjects::Count Method</td>
<td>161</td>
</tr>
<tr>
<td>IJSObjects::item Method</td>
<td>162</td>
</tr>
<tr>
<td>IJSObjects::Clear Method</td>
<td>162</td>
</tr>
<tr>
<td>IJSObjects::Add Method</td>
<td>162</td>
</tr>
<tr>
<td>IJSObjects::Remove Method</td>
<td>163</td>
</tr>
<tr>
<td>IJSProperty Interface</td>
<td>163</td>
</tr>
<tr>
<td>IJSProperty Methods</td>
<td>163</td>
</tr>
<tr>
<td>ReadOnly Method</td>
<td>164</td>
</tr>
</tbody>
</table>
Server Configuration Library

VirtualUIS::ProfileKind Enumeration .................................................. 191
VirtualUIS::ScreenResolution Enumeration ......................................... 191
VirtualUIS::ServerSection Enumeration ............................................. 192
IServer Interface .................................................................................. 192
IServer Methods .................................................................................. 193
IServer::Binding Method .................................................................... 193
IServer::Certificate Method ............................................................... 194
IServer::RDSCredentials Method ....................................................... 194
IServer::Profiles Method ................................................................ 195
IServer::Load Method ....................................................................... 195
IServer::Save Method ...................................................................... 195
IServer::HideSection Method ............................................................ 195
IServer::ShowSection Method ............................................................ 196
IServer::License Method .................................................................. 196
ILicense Interface ................................................................................ 197
ILicense Methods ............................................................................... 197
ILicense::Activate Method ............................................................... 197
ILicense::Deactivate Method ............................................................ 198
Properties .......................................................................................... 198
CustomerID ....................................................................................... 198
Limits ................................................................................................. 198
Features ............................................................................................ 198
IsTrial ................................................................................................. 199
SerialStr ........................................................................................... 199
IProfile Interface ................................................................................ 199
Properties ........................................................................................ 199
ID ....................................................................................................... 200
Name ................................................................................................. 200
VirtualPath ......................................................................................... 201
IsDefault ......................................................................................... 201
Enabled ......................................................................................... 201
ProfileKind ....................................................................................... 201
FileName ......................................................................................... 201

© 2015, Cybele Software, Inc.
2 .NET Classes .................................................................................................................. 211

Cybele.Thinfinity Namespace ....................................................................................... 211

Cybele.Thinfinity.BrowserOrientation Enumeration .............................................. 212
Cybele.Thinfinity.MouseMoveGestureStyle Enumeration ........................................... 213
Cybele.Thinfinity.MouseMoveGestureAction Enumeration ......................................... 213

ClientSettings Class .................................................................................................... 213
ClientSettings.ClientSettings Constructor .................................................................. 214
ClientSettings Methods ............................................................................................... 214
ClientSettings.Dispose Method ..................................................................................... 214

ClientSettings Properties ............................................................................................ 215
ClientSettings.MouseMoveGestureStyle Property ....................................................... 215
ClientSettings.MouseMoveGestureAction Property ................................................... 215
ClientSettings.CursorVisible Property ....................................................................... 215

BrowserInfo Class ....................................................................................................... 216
Contents

© 2015, Cybele Software, Inc.

UploadFile Method

ReceiveMessageArgs Class

CloseArgs Class

GetUploadDirEventArgs Class

BrowserResizeEventArgs Class

BrowserInfo Constructor........................................................................................................... 216
BrowserInfo Methods .................................................................................................................. 216
BrowserInfo.Dispose Method........................................................................................................ 217
BrowserInfo.GetCookie Method................................................................................................ 217
BrowserInfo.SetCookie Method................................................................................................ 217
BrowserInfo Properties .............................................................................................................. 217
BrowserInfo.ViewWidth Property............................................................................................... 218
BrowserInfo.ViewHeight Property.............................................................................................. 218
BrowserInfo.BrowserWidth Property......................................................................................... 219
BrowserInfo.BrowserHeight Property....................................................................................... 219
BrowserInfo.ScreenWidth Property.......................................................................................... 219
BrowserInfo.ScreenHeight Property......................................................................................... 219
BrowserInfo.Username Property............................................................................................... 220
BrowserInfo.IPAddress Property.............................................................................................. 220
BrowserInfo.UserAgent Property.............................................................................................. 220
BrowserInfo.UniqueBrowserId Property.................................................................................. 220
BrowserInfo.Location Property.................................................................................................. 220
BrowserInfo.ScreenResolution Property.................................................................................. 221
BrowserInfo.Orientation Property.............................................................................................. 221
DevServer Class ........................................................................................................................ 221
DevServer.DevServer Constructor............................................................................................. 222
DevServer Methods .................................................................................................................... 222
DevServer.Dispose Method.......................................................................................................... 222
DevServer Properties .................................................................................................................. 222
DevServer.Enabled Property........................................................................................................ 223
DevServer.Port Property............................................................................................................. 223
DevServer.StartBrowser Property.............................................................................................. 223
BrowserResizeEventArgs Properties............................................................................................ 223
BrowserResizeEventArgs.Width Property................................................................................... 224
BrowserResizeEventArgs.Height Property................................................................................ 224
BrowserResizeEventArgs.ResizeMaximized Property.................................................................. 224
GetUploadDirEventArgs Class.................................................................................................... 225
GetUploadDirEventArgs Properties............................................................................................ 225
GetUploadDirEventArgs.Directory Property.............................................................................. 226
GetUploadDirEventArgs.Handled Property............................................................................... 226
DownloadEndArgs Class............................................................................................................ 226
DownloadEndArgs Properties..................................................................................................... 227
DownloadEndArgs.Filename Property........................................................................................ 227
CloseArgs Class.......................................................................................................................... 227
ReceiveMessageArgs Class.......................................................................................................... 228
ReceiveMessageArgs Properties................................................................................................. 228
ReceiveMessageArgs.Data Property............................................................................................ 228
VirtualUI Class............................................................................................................................ 229
VirtualUI.VirtualUI Constructor.................................................................................................. 229
VirtualUI Methods....................................................................................................................... 229
DownloadFile Method................................................................................................................ 230
VirtualUI.DownloadFile Method (string).................................................................................. 230
VirtualUI.DownloadFile Method (string, string)...................................................................... 231
VirtualUI.DownloadFile Method (string, string, string).......................................................... 231
Start Method .............................................................................................................................. 231
VirtualUI.Start Method ()......................................................................................................... 232
VirtualUI.Start Method (int)...................................................................................................... 232
UploadFile Method..................................................................................................................... 232
<table>
<thead>
<tr>
<th>Cybele.Thinfinity.JSMethodCallback Type</th>
<th>249</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSCallback Class</td>
<td>249</td>
</tr>
<tr>
<td>JSCallback.JSCallback Constructor</td>
<td>250</td>
</tr>
<tr>
<td>JSCallback Methods</td>
<td>250</td>
</tr>
<tr>
<td>JSCallback.Dispose Method</td>
<td>250</td>
</tr>
<tr>
<td>JSCallback.Callback Method</td>
<td>251</td>
</tr>
</tbody>
</table>

**Cybele.Thinfinity.Settings.VirtualUI Namespace** | 251

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cybele.Thinfinity.Settings.VirtualUI.ProfileKind Enumeration</td>
<td>252</td>
</tr>
<tr>
<td>Cybele.Thinfinity.Settings.VirtualUI.ScreenResolution Enumeration</td>
<td>252</td>
</tr>
<tr>
<td>Server Class</td>
<td>253</td>
</tr>
<tr>
<td>Server.Server Constructor</td>
<td>254</td>
</tr>
<tr>
<td>Server.Methods</td>
<td>254</td>
</tr>
<tr>
<td>Server.Dispose Method</td>
<td>255</td>
</tr>
<tr>
<td>Server.Load Method</td>
<td>255</td>
</tr>
<tr>
<td>Server.Save Method</td>
<td>255</td>
</tr>
<tr>
<td>Server.HideSection Method</td>
<td>255</td>
</tr>
<tr>
<td>Server.Show Section Method</td>
<td>256</td>
</tr>
<tr>
<td>Server.Properties</td>
<td>256</td>
</tr>
<tr>
<td>Server.Binding Property</td>
<td>257</td>
</tr>
<tr>
<td>Server.Certificate Property</td>
<td>257</td>
</tr>
<tr>
<td>Server.RDSAccounts Property</td>
<td>257</td>
</tr>
<tr>
<td>Server.Profiles Property</td>
<td>258</td>
</tr>
<tr>
<td>Server.License Property</td>
<td>258</td>
</tr>
<tr>
<td>ServerUtils Class</td>
<td>258</td>
</tr>
<tr>
<td>ServerUtils.Methods</td>
<td>259</td>
</tr>
<tr>
<td>ServerUtils.RunAsAdmin Method</td>
<td>259</td>
</tr>
<tr>
<td>ServerUtils.Base64ToIcon Method</td>
<td>259</td>
</tr>
<tr>
<td>ServerUtils.IconToBase64 Method</td>
<td>260</td>
</tr>
</tbody>
</table>

**IRDSAccounts Interface** | 260

<table>
<thead>
<tr>
<th>IRDSAccounts Methods</th>
<th>261</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRDSAccounts.Add Method</td>
<td>261</td>
</tr>
<tr>
<td>IRDSAccounts.Delete Method</td>
<td>261</td>
</tr>
<tr>
<td>IRDSAccounts.Indexers</td>
<td>262</td>
</tr>
<tr>
<td>IRDSAccounts.this Indexer</td>
<td>262</td>
</tr>
<tr>
<td>IRDSAccounts.Properties</td>
<td>262</td>
</tr>
<tr>
<td>IRDSAccounts.Count Property</td>
<td>262</td>
</tr>
</tbody>
</table>

### 3 Delphi Classes

**VirtualUI_SDK Namespace** | 263

<table>
<thead>
<tr>
<th>TBrowserInfo Class</th>
<th>264</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBrowserInfo.Create Constructor</td>
<td>264</td>
</tr>
<tr>
<td>TBrowserInfo.Destroy Destructor</td>
<td>264</td>
</tr>
<tr>
<td>TBrowserInfo.Properties</td>
<td>265</td>
</tr>
<tr>
<td>TBrowserInfo.ViewWidth Property</td>
<td>265</td>
</tr>
<tr>
<td>TBrowserInfo.ViewHeight Property</td>
<td>266</td>
</tr>
<tr>
<td>TBrowserInfo.BrowserWidth Property</td>
<td>266</td>
</tr>
<tr>
<td>TBrowserInfo.BrowserHeight Property</td>
<td>266</td>
</tr>
<tr>
<td>TBrowserInfo.ScreenWidth Property</td>
<td>266</td>
</tr>
<tr>
<td>TBrowserInfo.ScreenHeight Property</td>
<td>266</td>
</tr>
<tr>
<td>TBrowserInfo.Username Property</td>
<td>267</td>
</tr>
<tr>
<td>TBrowserInfo.IPAddress Property</td>
<td>267</td>
</tr>
<tr>
<td>TBrowserInfo.UserAgent Property</td>
<td>267</td>
</tr>
</tbody>
</table>

© 2015, Cybele Software, Inc.
4 C++ Classes ......................................................................................................................................... 299

CClientSettings Class ............................................................................................................................ 299
CClientSettings::CClientSettings Constructor ................................................................................. 300
CClientSettings::~CClientSettings Destructor .................................................................................. 300
CClientSettings Methods ...................................................................................................................... 300
CursorVisible Method ......................................................................................................................... 301
CBrowserInfo Class

CBrowserInfo::CBrowserInfo Constructor ................................................................. 305
CBrowserInfo::~CBrowserInfo Destructor ................................................................. 306

CBrowserInfo Methods............................................................................................... 306

View Height Method .................................................................................................. 307
CBrowserInfo::ViewHeight Method ()....................................................................... 308
CBrowserInfo::ViewHeight Method (int) ................................................................... 308

View Width Method .................................................................................................... 308
CBrowserInfo::ViewWidth Method ()......................................................................... 308
CBrowserInfo::ViewWidth Method (int) .................................................................... 308

CBrowserInfo::getViewWidth Method ......................................................................... 309
CBrowserInfo::putViewWidth Method ......................................................................... 309

CBrowserInfo::getViewHeight Method ....................................................................... 309
CBrowserInfo::putViewHeight Method ...................................................................... 309

CBrowserInfo::getBrowserWidth Method ................................................................... 310
CBrowserInfo::getBrowserHeight Method ................................................................ 310

CBrowserInfo::getScreenWidth Method ...................................................................... 310
CBrowserInfo::getScreenHeight Method ................................................................... 310

CBrowserInfo::getUsername Method ......................................................................... 311

CBrowserInfo::getIPAddress Method ......................................................................... 311

CBrowserInfo::getUserAgent Method ......................................................................... 311
CBrowserInfo::getScreenResolution Method ............................................................. 312
CBrowserInfo::getOrientation Method ...................................................................... 312
CBrowserInfo::getUniqueId Method .......................................................................... 312

CBrowserInfo::GetCookie Method ............................................................................ 312
CBrowserInfo::SetCookie Method ............................................................................ 313

CBrowserInfo::getLocation Method ........................................................................... 313

CBrowserInfo::getBrowserWidth Method .................................................................. 313

CBrowserInfo::getBrowserHeight Method ................................................................. 313

CBrowserInfo::ScreenWidth Method .......................................................................... 314

CBrowserInfo::ScreenHeight Method ......................................................................... 314

CBrowserInfo::Username Method ............................................................................... 314

CBrowserInfo::IPAddress Method ............................................................................. 314

CBrowserInfo::UserAgent Method ............................................................................. 314

CBrowserInfo::UniqueId Method ................................................................................ 315

CBrowserInfo::Location Method ................................................................................ 315

CBrowserInfo::Orientation Method ........................................................................... 315

CBrowserInfo::getOrientation Method ...................................................................... 316

CBrowserInfo::setOrientation Method ...................................................................... 316

CBrowserInfo::getLocation Method ........................................................................... 316

CBrowserInfo::putLocation Method ........................................................................... 316

CBrowserInfo::UserAgent Method ............................................................................. 316

CBrowserInfo::UniqueId Method ................................................................................ 317

CBrowserInfo::Location Method ................................................................................ 317

CBrowserInfo::Orientation Method ........................................................................... 317

CBrowserInfo::setOrientation Method ...................................................................... 317

CBrowserInfo::getLocation Method ........................................................................... 317

CBrowserInfo::putLocation Method ........................................................................... 317

CBrowserInfo::UserAgent Method ............................................................................. 317

CBrowserInfo::UniqueId Method ................................................................................ 318

CBrowserInfo::Location Method ................................................................................ 318

CBrowserInfo::Orientation Method ........................................................................... 318

CBrowserInfo::setOrientation Method ...................................................................... 318

CBrowserInfo::getLocation Method ........................................................................... 318

CBrowserInfo::putLocation Method ........................................................................... 318

CBrowserInfo::UserAgent Method ............................................................................. 318

CBrowserInfo::UniqueId Method ................................................................................ 319

CBrowserInfo::Location Method ................................................................................ 319

CBrowserInfo::Orientation Method ........................................................................... 319

CBrowserInfo::setOrientation Method ...................................................................... 319

CBrowserInfo::getLocation Method ........................................................................... 319

CBrowserInfo::putLocation Method ........................................................................... 319

CBrowserInfo::UserAgent Method ............................................................................. 319

CBrowserInfo::UniqueId Method ................................................................................ 320

CBrowserInfo::Location Method ................................................................................ 320

CBrowserInfo::Orientation Method ........................................................................... 320

CBrowserInfo::setOrientation Method ...................................................................... 320

CBrowserInfo::getLocation Method ........................................................................... 320

CBrowserInfo::putLocation Method ........................................................................... 320

CBrowserInfo::UserAgent Method ............................................................................. 320

CBrowserInfo::UniqueId Method ................................................................................ 321

CBrowserInfo::Location Method ................................................................................ 321

CBrowserInfo::Orientation Method ........................................................................... 321

CBrowserInfo::setOrientation Method ...................................................................... 321
CDevServer Class ........................................................................................................................................... 316
CDevServer::CDevServer Constructor........................................................................................................... 316
CDevServer::~CDevServer Destructor................................................................................................................ 316
CDevServer Methods ........................................................................................................................................ 317
    Enabled Method........................................................................................................................................ 317
    CDevServer::Enabled Method () ................................................................................................................ 317
    CDevServer::Enabled Method (bool) ............................................................................................................. 318
    Port Method ............................................................................................................................................. 318
    CDevServer::Port Method () ....................................................................................................................... 318
    CDevServer::Port Method (int) ................................................................................................................... 318
    CDevServer::get_Enabled Method ................................................................................................................. 319
    CDevServer::put_Enabled Method ............................................................................................................... 319
    CDevServer::get_Port Method ...................................................................................................................... 319
    CDevServer::put_Port Method ...................................................................................................................... 319
    CDevServer::get_StartBrowser Method ..................................................................................................... 320
    CDevServer::put_StartBrowser Method ..................................................................................................... 320
    CDevServer::startBrowser Method ............................................................................................................... 320
JSObject Class .................................................................................................................................................. 320
JSObject::JSObject Constructor....................................................................................................................... 321
JSObject::~JSObject Destructor........................................................................................................................ 321
JSObject Methods ............................................................................................................................................. 322
    FireEvent Method..................................................................................................................................... 322
    JSObject::FireEvent Method (BSTR, LSArguments *)............................................................................... 323
    JSObject::FireEvent Method (std::string, LSArguments *)........................................................................ 323
    Id Method................................................................................................................................................. 323
    JSObject::Id Method () ............................................................................................................................. 324
    JSObject::QueryInterface Method.............................................................................................................. 324
    JSObject::get_Id Method............................................................................................................................. 324
    JSObject::put_Id Method............................................................................................................................. 324
    JSObject::get_Properties Method.............................................................................................................. 325
    JSObject::get_Methods Method................................................................................................................. 325
    JSObject::get_Events Method.................................................................................................................... 325
    JSObject::get_Objects Method.................................................................................................................... 325
    JSObject::ApplyChanges Method............................................................................................................... 326
    JSObject::ApplyModel Method.................................................................................................................... 326
    JSObject::Properties Method..................................................................................................................... 326
    JSObject::Methods Method.......................................................................................................................... 326
    JSObject::Events Method............................................................................................................................ 326
    JSObject::Objects Method............................................................................................................................ 326
JSCallback Class ............................................................................................................................................... 327
JSCallback::JSCallback Constructor.................................................................................................................. 327
JSCallback Methods ....................................................................................................................................... 328
JSCallback::Callback Method.......................................................................................................................... 328
JSBinding Class ............................................................................................................................................. 328
JSBinding::JSBinding Constructor................................................................................................................... 329
JSBinding Methods....................................................................................................................................... 329
    JSBinding::Set Method.............................................................................................................................. 329
VirtualUI ........................................................................................................................................................ 329
5 VirtualUI.sdk.min.js .................................................................................................................................. 330
Thinfinity.VirtualUI ......................................................................................................................................... 330
    Events ...................................................................................................................................................... 331
    onLoading ............................................................................................................................................. 331
    onShow .................................................................................................................................................. 331
onClose ................................................................................................................................. 332
onError ................................................................................................................................. 332
Properties ............................................................................................................................... 333
devMode .................................................................................................................................. 333

**Thinfinity.JsRO** .................................................................................................................. 333

Methods .................................................................................................................................. 334
start ........................................................................................................................................ 334
on ............................................................................................................................................ 334
off .......................................................................................................................................... 335

Properties .................................................................................................................................. 336
model ........................................................................................................................................ 336
Note to readers: this is the help file for Thinfinity VirtualUI v1.

Please visit www.cybelesoft.com/support/documentation.aspx for documentation on the latest version of this product.
1 Introduction

Thinfinity® VirtualUI is a software solution that enables developers to add user interface web remoting to their Windows applications. Typically, by adding one line of code to the application's project, you can transform your Windows Application into a cross-platform one, enabling it to run as usual on Windows or as a Web application when installed under a Thinfinity® VirtualUI Server environment.

Why Thinfinity® VirtualUI?

1. It enables you to effortlessly create dual-platform Windows/HTML5 Apps.

2. It expands applications availability by delivering them instantly to users anywhere on any device.

3. It dramatically reduces the Total Cost of Ownership (TCO), by slashing IT costs and simplifying administration avoiding costly virtualization/remoting solutions such as Citrix XenApp® or Microsoft® RemoteApp.

See more:

Architecture
Getting Started
Installing Thinfinity® VirtualUI
Compiling the application
Registering the application
Accessing the app from the Web
Thinfinity® VirtualUI Server Manager
Managing the SSL

Copyright © 2015, Cybele Software, Inc. All rights reserved.
2 Architecture

Thinfinity® VirtualUI™ is composed of:

- Thinfinity VirtualUI Javascript Client, running on an HTML5-capable web browser
- Thinfinity VirtualUI Server, running as a Windows Service
- Thinfinity VirtualUI SDK dlls and/or source code files libraries referenced and/or included in your Windows application
- Thinfinity VirtualUI Gateway, optional scaling and load balancing components.

Requirements:

Development machine

- Operating Systems:
  - Microsoft Windows 7 32bit / 64bit
  - Microsoft Windows 8 32bit / 64bit
  - Microsoft Windows Server 2012
- A development environment:
  - Microsoft Visual Studio
  - Delphi 5 to XE7
  - C++ Builder
  - Microsoft Visual Basic
  - PowerBuilder
  - other
- The application can use .NET WPF, GDI, GDI+ and limited DirectX calls.

Server and Gateway machine

- Windows 8 32-bit / 64-bit
- Windows Server 2012
End-user machine

- Any operating system and/or device with an HTML5-compliant web browser.
- Any modern web browser (HTML5-compliant) such as IE10/11, Chrome, Safari, Firefox, Opera, etc.
3 Getting Started

This section covers the basics of Thinfinity VirtualUI as follows:

1. Installing Thinfinity VirtualUI
2. Compiling and testing your WinForms, Delphi or C++ application
3. Registering the application in Thinfinity VirtualUI Server
4. Accessing the application from the Web

Find a more comprehensive reference here:

Development Server Manager
Advanced Programming with jsRO
Server Manager
Symbol Reference

3.1 Installing Thinfinity VirtualUI

1. Download the installer from the download page in the Cybele Software site:

   http://www.cybelesoft.com/downloads/

2. Execute the installer on the target machine.
3. Select the environments to be installed:

- Thinfinity® VirtualUI™ Server
  - Thinfinity® VirtualUI™ Server will be installed as a Windows™ service, ready to be used in a production environment.
  - Only available for Windows™ 8, Windows™ Server 2012 and later versions.

- Development Environment
  - The development environment will allow you to compile Windows™ applications with the Thinfinity® VirtualUI™ components, in order to make them accessible from HTML5 browsers.
**Thinfinity® VirtualUI™ Server**

Thinfinity® VirtualUI™ Server is an HTTPS/WebSocket Server that maintains the communication between your app and the web browser. Installing this environment will enable you to configure and run your adapted Windows apps. This is not needed for development purposes but its installation is recommended for testing purposes.

On this installation mode, the Thinfinity® VirtualUI™ Server will be installed as a Windows Service.

**Development Environment**

This environment is meant to be installed on the developer machine. This mode installs the Thinfinity® VirtualUI™ SDK files that you will need to include or reference in your application's project. It includes also one Thinfinity® VirtualUI™ Server that will execute in 'development' mode, to quickly test your application from a web browser.

4. If you are installing the Server Environment, you will be presented with the following options:

   ![Server Installation Options](image)

   **Server Installation Options**

   Choose your server mode setup options

   - Standard Mode
   - Load Balancing Mode

   - Gateway and Server roles
     - VirtualUI will be able to both receive and forward connections
   - Server Role
     - Receive connections from a Gateway Role installation
   - Gateway Role
     - Forward connections to a Server Role installation

   ![InstallShield Wizard](image)

**Standard Mode**

This is the default option. Choose this for a stand-alone installation, Thinfinity VirtualUI Server will centralize all the connections.
Load Balancing Mode

Choose this option to distribute the connection's load between several installations. Read more about Scaling and Load Balancing.

5. Press Next and wait for the installation process to finish. When it is done, press the 'Finish' button.

Read More:
- Simple UI Remoting

3.2 Simple UI Remoting

UI Remoting is made simple with Thinfinity VirtualUI. All it takes is adding one or two lines of code in your applications, compiling and then configuring VirtualUI Server to show it.

Read more:
- Compiling and testing a WinForms application
- Compiling and testing a Delphi application
- Compiling and Testing a C++ Application
- Registering the application in Thinfinity VirtualUI Server
- Accessing the app from the Web

© 2015, Cybele Software, Inc.
3.2.1 Compiling and Testing a WinForms Application

Follow these steps to integrate a .NET WinForms application with Thinfinity VirtualUI:

1. Open Microsoft Visual Studio.

2. Open the application's project you want to integrate with.

3. Right-click on the project name in the 'Solution Explorer' panel and then select 'Add' - 'Existing Item'. Look for the `Thinfinity.VirtualUI.cs` file, which is typically located in `c:\Program Files\Thinfinity\VirtualUI\Dev\dotNet`.

4. In the `program.cs` file add a line as follows:

```csharp
using System;
using System.Windows.Forms;

namespace MyApp
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            new Cybele.Thinfinity.VirtualUI().Start();
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

5. Compile your program and run it.

You will see this message. If you press the 'Start Web Browser' button, a web browser will open pointing to the application.
6. Alternatively, choose to not show the message and observe on the Windows Tray Bar that the Thinfinity VirtualUI Development Server icon will appear.

You can right-click on the icon, and after that click on the 'Open Web Browser' menu.

A Web browser window will open and your application will be running inside.

When you run your application under an IDE, it automatically starts in web mode. To access the application, open your web browser and point to http://127.0.0.1:6080.

8. You can also click on the 'Server Manager' menu of the Tray Bar icon to access the Development Server Manager.

Read more:

- Compiling and testing a Delphi application
- Compiling and Testing a C++ Application
- Registering the application in Thinfinity VirtualUI Server
- Accessing the app from the Web
- Application Execution behavior
3.2.2 Compiling and Testing a Delphi Application

Follow these steps to integrate a Delphi VCL application with Thinfinity VirtualUI:

1. Open Delphi.

2. Open the application's project.

3. Add the `VirtualUI_AutoRun` unit to the 'Uses' clause of the project source file. This unit will typically be located in the `C:\Program Files (x86)\Thinfinity\VirtualUI\dev\Delphi` folder.

```delphi
program MyApp;

uses
  Windows,
  Forms,
  VirtualUI_AutoRun,
  MyApp.Main in MyApp.Main.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

4. Compile the program and run it.

You will see this message. If you press the 'Start Web Browser' button, a web browser will open pointing to the application.

5. Alternatively, choose to not show the message and observe on the Windows
Tray Bar that the Thinfinity VirtualUI Development Server icon will appear.

6. Right-click on the icon, and after that click on the 'Open Web Browser' menu.

A Web browser window will open and your application will be running inside.

⚠️ When you run your application under an IDE, it automatically starts in web mode. To access the application, open your web browser and point to http://127.0.0.1:6080

7. Click on the 'Server Manager' menu to access the Development Server Manager.

Read more:
- Compiling and testing a WinForms application
- Compiling and Testing a C++ Application
- Registering the application in Thinfinity VirtualUI Server
- Accessing the app from the Web
- Application Execution behavior

3.2.3 Compiling and Testing a C++ Application

Follow these steps to integrate a C++ application with Thinfinity VirtualUI:


2. Open the application's project.

3. Add the path to the VirtualUI library in the 'Include Directories' field, which can be found in the main menu under Project - Properties - Configuration Properties - VC++ Directories).
4. Add Thinfinity.VirtualUI.cpp to the Source Files.
5. Add Thinfinity.VirtualUI.h to the Header Files.
6. Include Thinfinity.VirtualUI.h in the application`s cpp:
   
   ```cpp
   #include "Thinfinity.VirtualUI.h"
   ```
   
7. Create a VirtualUI instance of the program and start it.

   ```cpp
   BOOL CMFCTestApp::InitInstance()
   {
       INITCOMMONCONTROLSEX InitCtrls;
       InitCtrls.dwSize = sizeof(InitCtrls);
       InitCtrls.dwICC = ICC_WIN95_CLASSES;
       InitCommonControlsEx(&InitCtrls);
       CWinAppEx::InitInstance();
       if (!AfxOleInit())
       {
           AfxMessageBox(IDP_OLE_INIT_FAILED);
           return FALSE;
       }

       // Initialize VirtualUI
       VirtualUI vui;
       vui.Start();
   }
   ```
8. Compile the program and run it.

You will see this message. If you press the 'Start Web Browser' button, a web browser will open pointing to the application.

9. Alternatively, choose to not show the message and observe on the Windows Tray Bar that the Thinfinity VirtualUI Development Server icon will appear.

10. Right-click on the icon, and after that click on the 'Open Web Browser' menu.

A Web browser window will open and your application will be running inside.
access the application, open your web browser and point to http://127.0.0.1:6080

11. Click on the 'Server Manager' menu to access the Development Server Manager.

Read more:

- Compiling and testing a WinForms application
- Compiling and testing a Delphi application
- Registering the application in Thinfinity VirtualUI Server
- Accessing the app from the Web
- Application Execution behavior

3.2.4 Registering the Application in Thinfinity VirtualUI Server

Now the application needs to be registered on Thinfinity VirtualUI Server. By doing this, the application will get published on the web server and will be available to be run from the web.

We assume you have already Compiled and tested your application with the Thinfinity® VirtualUI™ runtime units.

To create an application profile, follow these steps:

1. Open the Thinfinity VirtualUI Server Manager, available in the Start Menu.
2. Go to the 'Applications' tab.
3. Click on the 'Add' button.
4. Name the application and inform the application path and file name.
5. You can check the 'Default Application' option to bypass the Thinfinity VirtualUI landing page and go directly to the selected profile. You can access the rest of the profiles through their virtual path urls.
6. Press 'OK' and 'Apply' on the Server Manager screen.

Now the application is ready to be reached on the Web.

Read More:
- Accessing the App from the Web
- Application Execution Behavior
- The 'Applications' tab
3.2.5 Accessing the App from the Web

Follow the next steps to access registered applications using the web browser:

1. Open your preferred web browser.

2. Type in the application URL. This URL is composed of the server URL plus the Virtual Path configured for the application, i.e. http://your-machine:6580/MyApp/

   Alternatively, leave the Virtual Path as the root path using just http://your-machine:6580/. In this case a page with the list of applications will show up, unless you have set a profile to be the Default Application, in which case you will be connected to the Default Application.

   a. Check the 'Open in a new browser window' option if you want the application to be opened in another tab.

   b. Click on the corresponding icon of the application you want to access.

Read More:
- Application Execution Behavior
3.2.6 Application Execution behavior

The application execution behavior will depend on how the application is run.

- **Windows Shell**
  When the application is executed from the Windows Shell, it will behave as a standard Windows application.

- **Development Environment**
  When the application is executed under a Development Environment (such as Microsoft Visual Studio or Embarcadero Delphi), an instance of VirtualUI Server running in development mode will be started, and the application will be seen both as an standard Windows application and as a Web application.

- **VirtualUI Server's web page:**
  If the application is launched from a VirtualUI Server's page, it will run as a web application.
3.3 Adapting the Application

Thinfinity VirtualUI not only exposes the original application on the web browser, it also allows you to integrate, extend it and customize its look using web resources.

Read more:
- Customizing the web page
- Programming VirtualUI
3.3.1 Customizing the Web Page

By default, the applications are loaded in the app.html page, located in the VirtualUI web directory. However, if you need to change the look and feel, add a new functionality, etc., you can achieve this by loading the application in a different web page with the use of a virtual path.

Read more:
- Preparing the web page
- Create a virtual path for the application
3.3.1.1 Preparing the Web Page

Create a new directory and the web page where the application will run. You can use app.html as a template. In order for the application to work in the browser, the page must have, at least:

1. A reference to the thinfinity.virtualui.css stylesheet.
2. A reference to the virtualui.sdk.min.js javascript library.
3. A div named 'virtualui' that will work as 'desktop' for the application.
4. The necessary code to create an instance of the Thinfinity.VirtualUI class, with a call to the connect() method.

For example:

```html
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> 
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"/>
  <meta name="apple-mobile-web-app-capable" content="yes" /> 
  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no, target-densityDpi=device-dpi"/>
  <title>Thinfinity VirtualUI - Basic Page</title>
  <link rel="stylesheet" type="text/css" href="css/thinfinity.virtualui.css" />
  <script src="virtualui.sdk.min.js" type="text/javascript"></script>
</head>
<body>
<script type="text/javascript">
  $(document).ready(function () {
    var virtualUI = new Thinfinity.VirtualUI();
    virtualUI.connect();
  });
</script>
<div id="virtualui" style="position:absolute;display:none;"></div>
</body>
</html>
```

Starting from this basic page, you can change the windows' style, add new html content and interact programmatically with the executed application.
**Thinfinity VirtualUI Server page load scheme**

In the previous example page you can see references to Thinfinity VirtualUI files (a stylesheet and a Javascript file). It's not necessary for those common files to be replicated in each virtual path folder. The server will handle each http request from a virtual path in the following order:

1) Search for the page locally, in the folder assigned to the application's virtual path.
2) Search for the page starting in the Thinfinity VirtualUI root directory.

**Read more:**
- [Customizing Settings](#)
3.3.1.2 Customizing Settings

After creating the directory and the web page, inform Thinfinity VirtualUI of their location by defining a virtual path for the application and indicating which page will be loaded by default.

In order to set a virtual path for the application:

1. Open the Thinfinity VirtualUI Server Manager.
2. Go to the 'Applications' tab.
3. Add the new application or select it from the list.
4. Click on the 'Edit' button.
5. Set the Home Page. Press the 'Open' button and search for the location of your customized home page.

6. You can modify the virtual path name.
7. Press 'OK' and 'Apply'.

When the application is loaded —through the icon in the index page or typing the url to the full virtual path—, Thinfinity VirtualUI will open the page that was
specified in the 'Home Page' field of the application profile, and it will show the application.

**Read More:**
- [Programming VirtualUI](#)
3.3.2 Programming VirtualUI

The virtualui-sdk.min.js javascript library has everything that you need to connect to your application and interact from it from the web page, extending and integrating with your execution environment as much as you wish.

The library has these classes:

- Thinfinity.VirtualUI class
- Thinfinity.JsRO class

**Thinfinity.VirtualUI class**

This class is necessary and mandatory, because it’s the one handling the communication between the browser and Thinfinity VirtualUI Server.

**Properties**

devMode

**Methods**

connect

**Events**

onClose
onError
onLoading
onShow

**Read More:**

- [Handling VirtualUI Events](#)
### 3.3.2.1 Handling VirtualUI Events

The following example shows how to assign *handlers* to events available in the VirtualUI object (they can be seen in the javascript console or shown as alerts in the case of errors or disconnections):

```html
<!DOCTYPE html>
<html>
<head>
  <title>Thinfinity VirtualUI - Basic Page</title>
  <link rel="stylesheet" type="text/css" href="css/thinfinity.virtualui.css" />
  <script src="virtualui.sdk.min.js" type="text/javascript"></script>
</head>
<body>
  <script type="text/javascript">
    $(document).ready(function () {
      var virtualUI = new Thinfinity.VirtualUI();

      virtualUI.onError = function (errorMsg) {
        if (errorMsg == null) { errorMsg = ""; }
        alert("Application load failed: " + errorMsg);
      };

      virtualUI.onLoading = function () {
        console.log("Loading...");
      };
      virtualUI.onShow = function () {
        console.log("Application loaded.");
      };
      virtualUI.onClose = function () {
        alert("Application closed.");
        window.close();
      };

      virtualUI.connect();
    });
  </script>
  <div id="virtualui" style="position:absolute;display:none;"></div>
</body>
</html>
```

**Thinfinity.jsRO class**

The Thinfinity.jsRO class helps create interaction between the application and the
web that's much more fluid than any other methodology or standard technology available.

This class interacts with the executable file and accesses the data models that are exposed from the application and the properties, methods and events that have been written for each of these models.

In the next chapter we will see how to work with jsRO (Javascript Remote Objects).

**Read More:**
- Advanced Programming with jsRO
4 Advanced Programming with jsRO

jsRO (JavaScript Remote Objects) allows you to increase your application's power, extending it to the web environment. This is made possible by the publishing of data models defined programmatically from the application that expose properties, methods and events, easing the dialog between the web and the original application.

To make the development of applications with jsRO easy, Thinfinity VirtualUI provides a web environment that allows you to interact with the application during the development and test its functionality: the Development Lab.

Read More:
- The Development Lab
- Life Cycle of jsRO Objects
- Creating jsRO Objects
- Updating Properties
- OnPropertyChange(), OnSet() y OnGet() events
- jsRO Remote Calls
4.1 The Development Lab

Thinfinity VirtualUI includes an html test environment called Development Lab. You can test your customized application and access the available programming resources through the objects. The Development Lab will load when you execute an application in Dev mode from your development environment by calling the URL where Thinfinity VirtualUI is running (i.e. http://127.0.0.1:6080/).

The environment consists of five panels:

- The Address bar.
  Load here the web page address that you want to open. If you call the root address ("/"), app.html will load; if you enter an application's virtual path, it will load the address that was specified as Home Page for this application in the Thinfinity VirtualUI Server Manager, and if the Home Page wasn't defined it will load app.html.

- The Model Inspector.
  The Model Inspector's function is to show the available properties, methods and events for each data model. In its upper part there's a combobox that shows the published models; by selecting an item you will see its attributes. From this panel you can change the properties value, provided that they are not read-only.

- The Log pane.
  The log pane shows a log of the interaction between the jsRO object and the application. The newer entrances will show on top.

- The Console pane.
  In the console pane you can assign a new value to a property or call a method.

- The Browser pane.
  This is the most important panel. The selected application will load there and it will show just like it would in the browser independently.

Read More:
- Life Cycle of jsRO Objects
- Creating jsRO Objects
- Updating Properties
- OnPropertyChange(), OnSet() y OnGet() events
- jsRO Remote Calls
4.2 Life cycle of the jsRO Objects

jsRO objects are always defined inside the application and propagated to the browser. New properties, methods or events can't be added from Javascript, but events that are produced can be handled, the property values can be retrieved and modified, and the object’s methods can be invoked.

There are four pre-defined Javascript events that are related to the life cycle of a jsRO object, associated to its creation, updating and destruction.

Starting from the first ApplyModel produced in the application, jsRO will create the object, propagate it to the browser and trigger there the event related to this creation:

```
on("model:object", "created")
```

also, for the properties that are created with an assigned value, jsRO will replicate their initial values in the browser and fire the corresponding events:

```
on("model:object.property", "changed")
```

and from that moment on, it will keep the state of the properties synchronized in the application and the browser.

If the model needs to be updated by the addition of new attributes (properties, events or method) a new ApplyModel must be made, which will, in turn, with the model already created, fire a changed event on the model:

```
on("model:object", "changed")
```

Finally, if the object is destroyed in the application, the deleted event will be triggered and the object will be destroyed in Javascript too.

```
on("model:object", "deleted")
```

This last event will also be triggered when closing the application.
Read More:
- Creating jsRO Objects
- Updating Properties
- OnPropertyChange(), OnSet() y OnGet() events
- jsRO Remote Calls
4.3 Creating jsRO objects

As stated previously, the sjRO models are created in the application and then propagated to the browser, where they can be consumed from Javascript through a Thinfinity.sjRO class instance.

Let's see a complete example of this sequence:

Both examples create a 'ro' object, which has a 'text' property with the value: 'Hello!'.

Using Delphi, in the Create method of the form:

```delphi
// Creates the remote object and its property
ro := TJSObject.Create('ro');
ro.Properties.Add('text');
ro.ApplyModel;
```

Using C# (.Net Winform application)

```csharp
// Creates the remote object and its property
ro = new JSObject("ro");
ro.Properties.Add("text");
ro.ApplyModel();
```

Let's see how to work with this object from Javascript:

```javascript
$(document).ready(function () {
    ... 
    ... 
    var jsro = new Thinfinity.JsRO();
    var ro = null;
    ... 
    ...
    jsro.on('model:ro', 'created', function () {
        ro = jsro.model.ro;
        });
    });
```

Sequence diagram for the creation of an object:
jsRO object creation

```
var jsro = new Thinfinity.JsRO();
var ro = null;
```

```
ro = new JObject("ro");
ro.Properties.Add("text");
ro.ApplyModel();
```

```
jsro.on('model.ro', 'created', function () { ro = jsro.model.ro; });
```

Read More:
- Updating Properties
- OnPropertyChange(), OnSet() y OnGet() events
- jsRO Remote Calls
4.4 Updating Properties

JsRO will keep each property synchronized in the browser and in the application. Each time a property is updated in the server side, this change triggers an event to the browser, where it can be handled by the corresponding `on("model:object.property", "change", ...)` event, if this event was declared. Similarly, when a property is updated in the browser, this change travels to the application and triggers the `object.OnPropertyChanged` and `property.OnSet` events.

Sequence diagram for the assignment of a property value from the application and from the browser:

Read More:
- `OnPropertyChanged()`, `OnSet()` y `OnGet()` events
- JsRO Remote Calls
4.5 OnPropertyChange(), OnSet() y OnGet() events

When publishing a model, jsRO will keep the state of the created object synchronized in the application side and in the browser. Each time a property is updated in the server side, this change is propagated to the browser, where besides updating the value it can be handled by the corresponding .on("model:object.property", "change", ...) event, if this event was declared. When a property is updated on the browser side, this change is sent in the opposite direction (from the web to the application) and triggers an OnPropertyChange event in the instantiated JSObject object. This is a good place to do things like propagating the change of a property value to some other element of the application, update a group of values in a database, etc.

jsRO can also handle changes in the properties of its object through the declaration of the OnSet and OnGet property events.

By invoking the ApplyChanges method on a JSObject object, the collection of the properties added to the object is traversed and if any of these properties has an OnGet event declared, it's triggered.

Using Property.OnGet
The OnSet event, however, is executed when it receives a change from a particular property from the browser.

**Using Property.OnSet**

```
// Creates the remote object
FRo := TJSObject.Create('ro');
// Property definition
FRo.Properties.Add('backgroundColor')
  .OnGet(TJSBinding.Create(
    procedure(const Parent: IJSObject; const Prop: IJSproperty)
    begin
      Prop.AsString := '#'
      + IntToHex(GetRValue(ColorToRGB(Form1.Color)), 2)
      + IntToHex(GetGValue(ColorToRGB(Form1.Color)), 2)
      + IntToHex(GetBValue(ColorToRGB(Form1.Color)), 2);
    end))
  .OnSet(TJSBinding.Create(
    procedure(const Parent: IJSObject; const Prop: IJSproperty)
    begin
      // Delphi code to set the background color from browser
    end))
```

The way to add the OnSet and OnGet event handlers to a property is based on their definition, you can do so when of adding the property or afterwards, always remember to fire an ApplyModel so that the model is propagated to the browser. The next example shows how the browser can retrieve the application form background color in a #RRGGBB format, and also how to change the background color using a value sent from the browser. Since the desktop application doesn't interpret colors like the web does, we need a conversion that works both ways:

Delphi Definition:

```
// Creates the remote object
FRo := TJSObject.Create('ro');
// Property definition
FRo.Properties.Add('backgroundColor')
  .OnGet(TJSBinding.Create(
    procedure(const Parent: IJSObject; const Prop: IJSproperty)
    begin
      Prop.AsString := '#'
      + IntToHex(GetRValue(ColorToRGB(Form1.Color)), 2)
      + IntToHex(GetGValue(ColorToRGB(Form1.Color)), 2)
      + IntToHex(GetBValue(ColorToRGB(Form1.Color)), 2);
    end))
  .OnSet(TJSBinding.Create(
    procedure(const Parent: IJSObject; const Prop: IJSproperty)
    begin
      // Delphi code to set the background color from browser
    end))
```
var
    value: string;
begin
    value := LowerCase(Prop.AsString);
    if ((Length(value) = 7) and (copy(value, 1, 1) = '#')) then
        try
            Form1.Color := RGB(
                StrToInt('$' + Copy(value, 2, 2)),
                StrToInt('$' + Copy(value, 4, 2)),
                StrToInt('$' + Copy(value, 6, 2))
            );
        except
        end;
end);

.Net definition:

    // Creates the remote object
    ro = new JSObject("ro");
    // Property definition
    ro.Properties.Add("backgroundColor")
        .OnGet(new JSBinding(
            // This anonymous procedure do the actual get
            delegate(IJSObject Parent, IJSProperty Prop)
            {
                Prop.AsString = "#"
                    + this.BackColor.R.ToString("X2")
                    + this.BackColor.G.ToString("X2")
                    + this.BackColor.B.ToString("X2");
            }));
    .OnSet(new JSBinding(
        // This anonymous procedure do the actual set
        delegate(IJSObject Parent, IJSProperty Prop)
        {
            string value = Prop.AsString.ToLower();
            Regex reColor = new Regex(@"^[0-9a-f]{6}\$");
            Match match = reColor.Match(value);
            if (match.Success)
                {
                    string color = match.Groups[1].Value;
                    this.BackColor = Color.FromArgb(
                        int.Parse(color.Substring(0, 2),
                        NumberStyles.AllowHexSpecifier),
                        int.Parse(color.Substring(2, 2),
                        NumberStyles.AllowHexSpecifier),
                        int.Parse(color.Substring(4, 2),
                        NumberStyles.AllowHexSpecifier)
                            );
            }
        }));
Assigning the property value in Javascript:

```
ro.backgroundColor = "#FF0000";
```
4.6 jsRO Remote Calls

In order to achieve more interaction between the remote application and the browser, the object model provided by jsRO allows the creation of remote methods and events defined in the application by the developer. Both the methods and the events are created and added to a model. While the methods can be invoked from the browser to remotely execute the application’s own actions, the events are fired from the application to the browser, where they can be handled by the corresponding callback.

Read More:
- Remote Methods
- Custom Events
4.6.1 Remote Methods

The remote jsRO methods allow you to make the application's own actions available to be invoked from Javascript. When the invocation of a method arrives to the application from the browser, two events are fired: the first is OnMethodExecuted, at the level of the JSObject object, which receives all the methods calls from the browser; the second is OnCall, and it happens at the level of the remote method. If the invoked method is a function (if it returns a value), the value will be returned and propagated to the browser, where it will be handled asynchronously by a callback defined as the last argument in the call to the method.

The following examples show how to add a method to an object and how it can be called from Javascript. In this case, we create a method called multiply, which will receive two integer type arguments as parameters and will return the result of the product between them. This result will be shown in the callback to the method call.

Method definition in Delphi:
// Creates the remote object
FRo := TJSObject.Create('ro');
// Adds the method
FRo.Methods.Add('multiply') // Returns a IJSMethod
 .AddArgument('a', JSDT_FLOAT) // First value to multiply
 .AddArgument('b', JSDT_FLOAT) // Second value to multiply
 .OnCall(TJSCallback.Create( // Adds the callback
     procedure(const Parent: IJSObject; const Method: IJSMethod)
     var
     a, b: int;
     begin
     a := Method.Arguments['a'].AsFloat;
     b := Method.Arguments['b'].AsFloat;
     Method.ReturnValue.AsFloat := a * b;
     end)
     .ReturnValue.DataType := JSDT_FLOAT; // Sets the return type
FRo.ApplyModel;

Method definition in .Net:

// Creates the remote object
ro = new JSObject("ro");
// Adds the method
ro.Methods.Add("multiply") // Returns a JSMethod
 .AddArgument("a", IJSDataType.JSDT_FLOAT) // First number to multiply
 .AddArgument("b", IJSDataType.JSDT_FLOAT) // Second number to multiply
 .OnCall(new JSCallback( // Adds the callback
delagate(IJSObject parent, IJSMethod Method)
{
    float a, b;
    a = Method.Arguments["a"].AsFloat;
    b = Method.Arguments["b"].AsFloat;
    Method.ReturnValue.AsFloat = a * b;
}).ReturnValue.DataType = IJSDataType.JSDT_FLOAT;
ro.ApplyModel();

Invoke the method in order to run it from Javascript. Use a callback in case the result needs to be retrieved (like in this case):

ro.multiply(3, 4, function (result) {
    alert("Result is " + result);
});
Read More:
• Custom Events
4.6.2 Custom Events

On top of the events detailed in Life Cycle of jsRO Objects, events can be created in the application that are defined by the programmer and when fired will be propagated from the application to the browser.

The following example shows how to add a personalized event to an object and how this can be handled from Javascript. In this case we’ll expose, as a JSON, the mouse coordinates.

Definition and use of an event in Delphi:

```delphi
// Creates the remote object
FRo := TJSObject.Create('ro');
// Adds the event
FRo.Events.Add('mousePositionChanged')
  .AddArgument('coords', JSDT_JSON); // Adds the mouse position as JSON
FRo.ApplyModel;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState;
                                 X, Y: Integer);
  begin
```
Definition and use of an event in .Net:

```csharp
// Creates the remote object
ro = new JSObject("ro");
// Adds the event
ro.Events.Add("mousePositionChanged")
  .AddArgument("coords", IJSDataType.JSDT_JSON) // Adds the mouse position as JSON
ro.ApplyModel();
...
...

private void Form1_MouseMove(object sender, MouseEventArgs e)
{
  ro.Events["mousePositionChanged"]
    .ArgumentAsJSON("coords",
      "{ "x": " + MousePosition.X + ", "y": " + MousePosition.Y + "}")
    .Fire();
}
```

To handle the event in Javascript, please note that the Javascript syntax for the personalized events differs slightly from the syntax for the model events, since it's declared directly associated to the name of the jsRO (without the "model"):

```javascript
jsro.on('ro', 'mousePositionChanged', function (coords) {
  console.log('mouse moved to [' + coords.x + ', ' + coords.y + ']');
});
```
5 Scaling and Load Balancing

Scaling and load balancing come into play when one machine is not capable of managing all the required resources. Too many concurrent connections or virtualized application that handle a lot of graphics, sound or other elements that require a great availability of resources may cause an overload.

Thinfinity VirtualUI provides components that allow you to distribute the workload across multiple Windows sessions, as well as multiple servers. You can scale the application availability in terms of applications instances —and user accesses— and failover scenarios in order to achieve optimal resource utilization and avoid overload.

Some of the benefits of load balancing:

- Avoids the overload by distributing the connections among different servers
- Minimizes response time
- More reliability (redundancy)
- Failover control

Read More:
- Scaling and Load Balancing Configurations
- Installing components
- Enabling Multiple RDS Accounts
- Configuring Load Balancing

5.1 Scaling and Load Balancing Configurations

If you arrive to the conclusion that your Thinfinity® VirtualUI environment would benefit from using Load Balancing, you can choose between different scenarios. This decision is an essential step in planning the hardware scheme and configuring the system to work in a distributed way.

Scenario 1: Multiple RDS accounts
This architecture involves only one computer. Each RDS session creates a Server instance which can, in turn, handle application instances separately. The VirtualUI Broker administrates the server instances: checks up on them to see if they are functional, and works together with the Gateway to distribute the connections.

### Scenario 2: Multiple Servers with Load Balancing

In this simple scenario, a single Gateway distributes the connection load between a number of Servers.
**Scenario 3: Multiple Gateways and Servers with Load Balancing**

Thinfinity® VirtualUI™ Load Balancing  
DNS Load Balancer + Multiple VirtualUI Gateways

This example combines a external load balancing DNS Server with multiple Gateways. The scheme is composed by multiple Servers, multiple Gateways and the DNS Server, its domain name associated to all the available Gateways' IPs.

**Scenario 4: Multiple RDS accounts and multiple Servers with Load Balancing**

Combine load balancing with multiple RDS sessions to get the most out of your architecture.

**Read More:**
- Installing components  
- Enabling Multiple RDS Accounts  
- Configuring Load Balancing
5.2 Installing components

In this section you will learn how to set up Thinfinity VirtualUI's components in a load-balancing network configuration.

Choose the Load Balancing mode in the Server Installation Options screen:

Two or more servers will participate in the load balancing/fault-tolerance scenario. These are the two possible roles for an installation:

**Gateway Role:** Under this role, VirtualUI responds to all web-page requests and, when a connection is solicited, it selects the appropriate Server to forward that request to.
In case any established connection fails, or a Server falls down, the Gateway will be able to reconnect to the Server that has the highest availability at the moment.
All the system settings and profiles are centralized and shared between the Servers.

**Server Role:** Under this role, VirtualUI only processes forwarded connections. The Server is responsible for establishing and processing the connections assigned by the Gateway. A Server Role installation also includes the option to enable multiple RDS accounts.

Before configuring a distributed environment, you should go over some steps:
1. Choose out of the possible **Scaling and Load Balancing Configurations** the one that best fits your needs.
2. Plan which machines will work under the Server Role, and which under the Gateway Role and DNS Servers.
3. Make sure all the Gateway Roles IP addresses are public to the web browsers that will access Thinfinity® VirtualUI.

**Read More:**
- [Enabling Multiple RDS Accounts](#)
- [Configuring Load Balancing](#)

### 5.3 Enabling Multiple RDS Accounts

In order to enable multiple RDS accounts, go to **the ‘RDS’ tab** and press the ‘Add’ button:

You can use already existing accounts or new VirtualUI accounts which will be created automatically.
The accounts must have administrator permissions.
Each RDS session will handle independent application instances, allowing a single computer to handle more connections.
Note: In order to use this feature, Thinfinity VirtualUI must be installed in a Windows Server where the Remote Desktop Session Host (RD Session Host) role service is installed.

Read More:
- Scaling and Load Balancing
5.4 Configuring Load Balancing

In order to configure a load balancing scenario, you need at least one Gateway installation and two Server installations.

**Configuring the Gateway Roles**

Under this role, VirtualUI responds to all web-page requests and, when a connection is solicited, it selects the appropriate Server to forward that request to.

To configure the Gateway, open the Gateway Manager. Set the IP and port where the Gateway will run. If you only have one gateway, this is where the users will connect to. If you use more than one Gateway in your architecture, you will use this IP in the DNS server you set up to distribute the connection between the Gateways.

Also, set the Network ID. All the Gateway and Server installations involved in a Load Balancing architecture share the same network ID.

---

**Configuring the Server Roles**
Under this role, VirtualUI only processes forwarded connections. The Server is responsible for establishing and processing the connections assigned by the Gateway.

To configure the Server, open the Server Manager and go the 'Gateways' tab. Press the 'Add' button to add a gateway to the Gateway List.

This means that now this server's resources can be accessed through the listed gateways.

Make sure that the Network ID is the same for all the gateways and servers involved in this load balancing architecture.

Then, go to the 'Applications' tab:
Set the 'Database Path' field in a network location that you can access from the other Server role installations. Once you share the database path, all the information in the 'Applications' tab will be shared with other VirtualUI installations. Make sure you modify the Applications information from one installation at a time, as all changes will be reflected in the other installations.

Also, make sure all the Gateways' IPs are public to the locations that will access Thinfinity® VirtualUI through a web browser.

**Read More:**
- The Gateway Manager
- Scaling and Load Balancing Configurations
- Configuring the General tab
- Configuring the RDS tab
- Configuring the Applications tab
- Configuring the Licenses tab
6 Enhanced Browser and DPI Support

Among the wide range of valid resolutions that Thinfinity VirtualUI offers, the most commonly used—for its flexibility and simplicity—is “Fit to Browser”. This configuration allows you to adjust the remote application to fit the available browser size. However, when it comes to accessing a desktop from different devices, the sometimes huge differences between screen sizes and pixel resolutions (i.e. iPhone 4 vs a 27 inch iMac Retina Display) make it impossible to have a simple rule to determine the best remote desktop size. Even when the application is adjusting properly to the available size, the screen rendered might still look tiny or disproportionate, making the user experience not as satisfactory as expected.

Tailoring "Fit to browser"

Now, using a new configurable browser detection ruleset, we can tailor the way we want to see of the remote desktop/application on every device. This ruleset allows you to specify rules that will detect the web browser, device and display characteristics, and set parameters that adjust the remote desktop/application resolution according to your own taste.

The main characteristics that need to be taken into account are:

- The browser User Agent, that tells about the web browser and device
- The device pixel ratio, that tells about the real display resolution
- The device display size
- The display orientation (landscape or portrait)

The browser detection ruleset is stored in a file with entries that contain specifications (rules) that match general or specific devices. Each entry (model) can inherit matching parameters (properties) from a more general model. For example, you can define an iOS model and an iPhone4 can inherit the iOS model properties.

A default ruleset file named BrowserRules.ini is installed in the Thinfinity VirtualUI program folder. Then, if it doesn’t exist there yet, it is copied to “\programData\Cybele Software\Thinfinity\VirtualUI” and renamed as Thinfinity.VirtualUI.BrowserRules.ini. You can safely customize this file as it won’t be overridden with a program update.

The structure of this file is as follow:

```ini
[default]
min-width = 640
min-height = 480
max-width = 2560
max-height = 1600
max-device-pixel-ratio = 1

[mobile]
parent-model = default
match-mobile = true
max-device-pixel-ratio = 2
```

Note: for these setting to apply, the connection’s ‘Resolution’ property must be set to ‘Fit to browser’.

Configure this setting in for your application in the 'General' tab of the Application Profile, or the...
A section defines a model, and each model contains a set of properties divided in two groups: matching properties and applying properties.

Models are organized in an inheritance tree. The relationship between models is defined by a special property rule called parent-model, present in all models except in the [default] model, which is the tree's root node and includes some basic properties.

Every other model must directly or indirectly inherit from the [default] model. Also, each model contains its own rules that match general or specific devices, and inherits all specifications (including matching parameters) from its ancestors.

When more than one criteria is met for a device, a scoring system is used to resolve this conflict.

This is the in-the-box models tree:
Properties can be divided in two groups: matching properties and applying properties.

Matching properties are those used to test the browser and device properties (such as the browser user agent, the device pixel ratio, the display orientation width and height, etc.) in order to choose the best model for each case.
<table>
<thead>
<tr>
<th><strong>match-device-pixel-ratio</strong></th>
<th>Matches any device with a specific pixel ratio.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>match-mobile</strong></td>
<td>Matches any mobile device.</td>
</tr>
<tr>
<td><strong>match-orientation</strong></td>
<td>Matches any device with the specified orientation: landscape or portrait.</td>
</tr>
<tr>
<td><strong>match-screen-height-range</strong></td>
<td>Matches any device with a screen height in the specified range. This range is expressed as From-To (for example, 900-1200).</td>
</tr>
<tr>
<td><strong>match-screen-width-range</strong></td>
<td>Matches any device with a screen width in the specified range. This range is expressed as From-To (for example, 400-600).</td>
</tr>
<tr>
<td><strong>match-screen-height</strong></td>
<td>Matches any device with a specified screen height.</td>
</tr>
<tr>
<td><strong>match-screen-width</strong></td>
<td>Matches any device with a specified screen width.</td>
</tr>
<tr>
<td><strong>match-user-agent</strong></td>
<td>Matches devices by comparing the device browser user agent to the string value supplied. This string is a regular expression.</td>
</tr>
</tbody>
</table>

Applying properties are those used to determine the final size and resolution.

Use the **parent-model** property to set the parent model:

<table>
<thead>
<tr>
<th><strong>parent-model</strong></th>
<th>Establish the parent model for this model.</th>
</tr>
</thead>
</table>

The following properties deal with the display resolution:

<table>
<thead>
<tr>
<th><strong>device-pixel-ratio</strong></th>
<th>Overrides the original device pixel ratio, scaling the content accordingly.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>max-device-pixel-ratio</strong></td>
<td>This property determines the maximum device pixel ratio</td>
</tr>
</tbody>
</table>
### pixel-ratio

accepted. The lesser of the device's device pixel ratio and this value is applied to scale the display.

The following properties deal with the screen size of the remote desktop, in pixels. You can determine it by setting the actual height and width, or by establishing maximum and minimum values for these properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>height</td>
<td>Remote desktop height.</td>
</tr>
<tr>
<td>width</td>
<td>Remote desktop width.</td>
</tr>
<tr>
<td>max-height</td>
<td>Remote desktop maximum height.</td>
</tr>
<tr>
<td>max-width</td>
<td>Remote desktop maximum width.</td>
</tr>
<tr>
<td>min-height</td>
<td>Remote desktop minimum height.</td>
</tr>
<tr>
<td>min-width</td>
<td>Remote desktop minimum width.</td>
</tr>
</tbody>
</table>

The following properties allow you to specify device screen areas that will never be used for displaying the remote connection, such as when a browser or device bar cannot be hidden and uses up screen space. These margins will be excluded for screen size calculations.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>margin-left</td>
<td>Width of an area at the left of the device screen that will not be used for displaying the remote desktop.</td>
</tr>
<tr>
<td>margin-bottom</td>
<td>Width of an area at the bottom of the device screen that will not be used for displaying the connection.</td>
</tr>
<tr>
<td>margin-right</td>
<td>Width of an area at the right of the device screen that will not be used for displaying the connection.</td>
</tr>
<tr>
<td>margin-top</td>
<td>Width of an area at the top of the device screen that will not be used for displaying the connection.</td>
</tr>
</tbody>
</table>

Miscellaneous properties:
use-full-screen

For mobile only. If the device’s browser supports the full-screen mode, this property indicates the remote desktop size should be calculated to occupy the whole screen. When not in full screen, the content will be scaled.

Read More:
- The Calculation Process
- Examples

6.3 The Calculation Process

In order to choose a model from the ruleset, Thinfinity uses the client device type, dimensions, resolution, orientation and browser:

1. If match-mobile exists, it tests if device is a mobile.
2. If match-user-agent exists, it tests the browser’s User Agent.
3. If match-device-pixel-ratio exists, it tests the device’s pixel ratio.
4. If match-orientation exists, it tests the device’s orientation.
5. If match-screen-width-range or match-screen-height-range exist, it tests to see if the screen size is in range.
6. If match-screen-width or match-screen-height exist, it tests the exact screen size.

Once the model is selected, the parameters are applied in this way:

1. If the width and height properties exist, then it applies them.
2. If the browser width is less than the min-width, it applies min-width.
3. If the browser height is less than the min-height, it applies min-height.
4. If the browser width is greater than the max-width, it applies max-width.
5. If the browser height is greater than the max-height, it applies max-height.
6. If a specific device-pixel-ratio was specified, it applies it.
7. If a max-device-ratio was specified, it takes the minimum of the real device pixel ratio and max-device-ratio property and applies it.

Read More:
- Examples

6.4 Examples

This example shows a possible ruleset and how it will affect different devices:

```
[default]
min-width = 640
min-height = 480
max-width = 2560
max-height = 1600
max-device-pixel-ratio = 1
```
In this case, when connecting with an ipad, the following models will be matched:

- [default]: This model applies to all devices.
- [mobile]: The ipad will match the match-mobile property.
- [ipad]: The ipad will match the user agent keyword 'ipad' specified in the match-user-agent property.

The resulting properties for this device will be:

- \( \text{min-width} = 640 \)
- \( \text{min-height} = 480 \)
- \( \text{max-width} = 2560 \)
- \( \text{max-height} = 1600 \)
- \( \text{max-device-pixel-ratio} = 2 \)

Using the same ruleset, when connecting with an iphone4, the following models will be matched:

- [default]: This model applies to all devices.
- [mobile]: The iphone will match the match-mobile property.
- [iphone4]: The iphone will match the user agent keyword 'iphone' specified in the match-user-agent property, together with the match-screen-width and match-screen-height properties. An iphone6, with a screen width of 667px, and a screen height of 375px, would match the 'iphone' user agent keyword, but not the size.

The resulting properties for this device will be:

- \( \text{min-width} = 640 \)
- \( \text{min-height} = 480 \)
- \( \text{max-width} = 2560 \)
- \( \text{max-height} = 1600 \)
- \( \text{max-device-pixel-ratio} = 2 \)

\( \text{device-pixel-ratio} = 1.5 \)

# One-Time URL

Thinfinity VirtualUI offers a special access method called “One-Time URL”. This mechanism was designed to create a temporary, unique url to provide one-time access to a specific application. This temporary url is disposed as soon as it is used or after a specified period of time has elapsed.

These are the main scenarios where the One-Time URL access method is most useful:
• Single Sign-on scenarios.
• External authentication methods.
• One-time invitations to run a program (i.e. application demos/presentations).

Read more:
• How it Works
• Creating a One-Time URL
• Single Sign-On Sequence Example

7.1 How it Works

The One-Time URL is an unique, disposable URL leading to a specific VirtualUI application. What makes it useful is that it allows for passing credentials and/or custom data to the application through an independent secure channel, hidden to the end user.
Server to request the creation of a One-Time URL, passing information about the application to run, credentials and custom data. This information is stored temporarily and indexed by a unique access key. Also a random passcode is created and used to encrypt the stored information. This access key and passcode are returned to the backend service to build the final One-Time URL.

Once the user is directed to the provided URL (automatically or by clicking on a link), VirtualUI validates the access key and passcode and starts the application passing the associated data. Finally, this key and associated data are removed from memory and therefore the URL becomes invalid. The same happens if the URL was not used for the amount of time specified in the creation request.

Read more:
- Creating a One-Time URL
- Single Sign-On Sequence Example

7.2 Creating a One-Time URL

The VirtualUI Server processes a One-Time URL creation request in the form of an http(s) request, as follows:

```
serverurl + "/ws/oturl/get?apikey=" + apikey + "&accesskey=" + accesskey + 
"&userid=" + userid + "&password=" + password + 
"&customdata=" + customData + 
"&plen=" + passlen + "&expires=" + expires,
```

where:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required/Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serverurl</td>
<td>optional</td>
<td>VirtualUI Server address (protocol, domain and port)</td>
</tr>
<tr>
<td>apikey</td>
<td>required</td>
<td>VirtualUI installation [API] key. Find this information in Thinfinity.VirtualUI.Server.ini at C:\ProgramData\Cybele Software\Thinfinity \VirtualUI</td>
</tr>
<tr>
<td>accesskey</td>
<td>required</td>
<td>Identifies the application that will be run. Complete this parameter with the 'access key' parameter found in the application's profile in the VirtualUI Server Manager.</td>
</tr>
<tr>
<td>userid</td>
<td>optional</td>
<td>A valid user that meets the criteria set in the application profile's 'Permission' tab.</td>
</tr>
<tr>
<td>password</td>
<td>optional</td>
<td>The password of the user specified in the 'userid' parameter.</td>
</tr>
</tbody>
</table>
customdata | optional | Use this field to send any information you may need to make available to the application. This is the right place to pass sign-on credentials. This information will be accessible in the application through the `BrowserInfo.CustomData` property.

| plen | optional | Length of the passcode to be returned.
| expires | optional | Ticket expiration time, in minutes.

**Note:** The user indicated in the 'userid' parameter is a user validated within VirtualUI to access the application. This is an Active Directory user that was either added individually, or as part of a group of users in the 'Permissions' tab of the profile that will be accessed. In a regular non-one-time-URL implementation, the end user would authenticate by providing these credentials. However, when using one-time-URL, this information is used internally by the application and the user can be authenticated somehow else.

If the request is unsuccessful, the following HTTP codes can be received:

400: Invalid parameters
401: Userid/password invalid

If the request is successful, the HTTP call returns a 200 HTTP status code, and a JSON consisting of two fields:
```
{
    "key": "LnJwsxGhp5d@6MHeiEswRdfxFCiIcLAUttRS$9FSUs-Utz3o",
    "pass": "1U4KRLN0"
}
```

With this information, the backend can build the final URL, following this format:
```
http(s)://server-url/oturl.html?key=[accesskey]&pass=[passcode]
```

Here’s an example that uses the JSON shown above:
```
http(s)://server-url/oturl.html?key=LnJwsxGhp5d@6MHeiEswRdfxFCiIcLAUttRS$9FSUs-Utz3o&pass=1U4KRLN0
```

**Read more:**
- Single Sign-On Sequence Example

### 7.3 Single Sign-On Sequence Example

The sequence diagram below shows a complete one-time-URL example using single sign-on. In this example, the same credentials —the 'jdoe' username and 'pass' password— are first used for the single sign-on, and then sent to the application in the `customdata` parameter.
In conclusion, the One-Time URL offers a useful way to extend web-enhanced applications to new scenarios. We are fully convinced that you will greatly benefit from this new Thinfinity VirtualUI feature.

8 Configuration Reference

This section is a reference for the different managing tools included in Thinfinity VirtualUI

Read More:
- The Development Server
- The Production Server
- The Gateway Manager

8.1 Development Server

Thinfinity® VirtualUI™ Developer Manager is a tool to manage your applications in development mode. Access the virtual path for each application to test your web developments.

When in production mode, you will use a very similar tool called Thinfinity VirtualUI Server Manager.

To access Thinfinity® VirtualUI™ Development Server Manager compile your application and run it. The Thinfinity VirtualUI Development Server icon will appear in the Windows Tray Bar.
Right-click on the icon, and after that click on the 'Open Web Browser' menu.

**Read More:**
- The 'General' Tab
- The 'Applications' Tab
- The 'Licenses' Tab
8.1.1 General

In the Thinfinity® VirtualUI™ Development Server Manager 'General' tab you will find the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bind to IP</td>
<td>Use this option to restrict access to the service to one specific IP address. The 'All unassigned' option allows access through all the available IP addresses.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Choose between the http and https protocol.</td>
</tr>
<tr>
<td>Port</td>
<td>Choose which port will Thinfinity® VirtualUI™ Server be listening on. If the port is not available, you will see an error message on the status bar.</td>
</tr>
</tbody>
</table>
8.1.2 Applications

In the Thinfinity® VirtualUI™ Development Server Manager Applications tab you will find the following options:

<table>
<thead>
<tr>
<th>Application List</th>
</tr>
</thead>
<tbody>
<tr>
<td>This list shows the available applications. You can enable or disable them by checking the box to the left of the name.</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Target</td>
</tr>
</tbody>
</table>

| Add | Press this button to add a new application. |
| Edit | Select an application and press this button to edit it. |
| Remove | Select an application and press this button to remove it. |
| Allowed users and groups for selected profile | See here the allowed users or group(s) of users for the selected application. If you want to change the permissions, edit the application. |
| Database path | Path to the profile database. |

**Read More:**
- [The Application Profile](#)
- [The Weblink Profile](#)
- [The 'Licenses' Tab](#)

### 8.1.2.1 Application Profile

When you edit or add an application profile you will be presented with this screen below.
The radio button 'Application' must be checked.
These are the profile properties you can edit:

<table>
<thead>
<tr>
<th>Name</th>
<th>Use this field to change the application name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Path</td>
<td>The Virtual Path will create a unique URL address for this connection. The complete path will consist of: http(s)://ip:port/VirtualPath/. Use this path and the home page field to test the look of the application.</td>
</tr>
<tr>
<td>Home Page</td>
<td>Choose the landing HTML page for the application and use it with the virtual path field to test its look.</td>
</tr>
<tr>
<td>Open</td>
<td>Press this button to look for the Home Page.</td>
</tr>
<tr>
<td>Access Key</td>
<td>This is a unique key for this application profile. The value is used to identify the application when implementing access through the One-Time-URL</td>
</tr>
</tbody>
</table>
**Icon**
Click on the Icon gray box to load an image to be associated with the profile. The image will be presented along with the profile name on the web interface profiles selection.

**Application/Web link**
Select the Application option to have a regular profile that gives access to an application. If you select the Web link radio button, this profile will behave like a Web Hyperlink.

**Default Application**
Check this option to make this profile the default application: the authenticated user will connect to this profile directly instead of choosing between the available profiles. The rest of the profiles can be accessed by their Virtual Path.

The properties located inside the tabs will be described throughout the next subtopics.

**Read More:**
- The 'General' Tab
- The 'Credentials' Tab
- The 'Permissions' Tab
- The Weblink Profile
8.1.2.1.1 General

In the Application Profile Editor 'General' tab you will find the following options:

<table>
<thead>
<tr>
<th>General</th>
<th>Credentials</th>
<th>Permissions</th>
</tr>
</thead>
</table>

- **Program path and file name**: Specify the complete path that gives access to the application executable file.
- **Arguments**: Application arguments.
- **Start in the following folder**: Inform a context directory for the application set on the 'Program path and file name' field.
- **Resolution**: Choose from the available list of resolutions including 'Fit to browser window' and 'Fit to screen', ideal for hiding the browser and working on a full screen mode.
- **Idle Timeout**: Set a timeout in minutes if you want Thinfinity® VirtualUI™ Server to wait this period before killing the application once the browser has been closed. Timeout 0 will kill the application immediately after the browser has been closed.

**Read More:**
- The 'Credentials' Tab
- The 'Permissions' Tab
### 8.1.2.1.2 Credentials

In the Thinfinity® VirtualUI™ Application Editor 'Credentials' tab, you should inform the mode for logging into the specified application:

<table>
<thead>
<tr>
<th>Use the authenticated credentials</th>
<th>Use the same credentials entered in the browser for Thinfinity® VirtualUI™ (specified in the 'Permissions' tab). Note: If the credentials are correct, this option will connect the user automatically when selecting the application, or after authenticating for Thinfinity® VirtualUI™ if this is the only profile for their credentials.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use these credentials</td>
<td>Complete the credentials used to access the computer. Note: If the credentials are correct, this option will connect the user automatically when selecting the application, or after authenticating for Thinfinity® VirtualUI™ if this is the only profile for their credentials.</td>
</tr>
</tbody>
</table>

Read More:
- [The 'Permissions' Tab](#)
8.1.2.1.3 Permissions

Select the users that will access this application. If you don't select any user, this application will not be accessed. These are the options you will find on the Application Profile Editor 'Permissions' tab:

- **Allow anonymous access**: Check this option to make this application available without any authentication. Use this option if you want this profile to be available for everyone. This means that everybody accessing Thinfinity® VirtualUI™ will have access to this application. Checking this option will disable the Add and Remove buttons.

- **Add**: Press 'Add' to access the windows dialog for selecting Active Directory users.

- **Remove**: Press 'Remove' to remove a user for this profile.

If you want a user or a user group to access more than one application, you need to create more application profiles and then add this user to each profile. The authenticated user will be able to choose from the available application profiles on the Web interface.

**Read More:**
- The Weblink Profile
8.1.2.2 Weblink Profile

When you edit or add a Web Link profile you will be presented with this screen below. The radio button 'Web Link' must be checked.

These are the profile properties you can edit:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Use this field to change the profile name.</td>
</tr>
<tr>
<td>Virtual Path</td>
<td>The Virtual Path will create a unique URL address for this connection. The complete path will consist of: http(s)://ThinfinityVirtualUIDomain:port/VirtualPath/. The users can then create a web shortcut to this connection in particular and bypass the Thinfinity® VirtualUI web interface.</td>
</tr>
<tr>
<td>Access Key</td>
<td>This is a unique key for this application profile. The value is used to identify the application when implementing access through the One-Time-URL method.</td>
</tr>
<tr>
<td>Icon</td>
<td>Click on the Icon gray box to load an image to be associated with the application. The image will be presented along with the application name on the web interface.</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Application / Web Link</td>
<td>Select the Weblink option to have a profile that connects to a Web link. These links will be shown along with all the other applications on the Thinfinity® VirtualUI™ start page.</td>
</tr>
<tr>
<td>Default Application</td>
<td>Check this option to make this profile the default application: the authenticated user will connect to this profile directly instead of choosing between the available profiles. The rest of the profiles can be accessed by their Virtual Path.</td>
</tr>
<tr>
<td>Web URL</td>
<td>Inform in this field the URL that you want this application profile to connect to.</td>
</tr>
</tbody>
</table>

The properties located inside the other tabs will be described throughout the next subtopics.

**Read More:**
- [The 'Permissions' Tab](#)
8.1.2.2.1 Permissions

Select the users that will access this application profile. If you don't select any users, this profile will not be available from the Web interface. These are the options you will find under the 'Permissions' tab:

- **Allow anonymous access**: Check this option to make this application available without any authentication. Use this option, if you want this profile to be available for everyone. This means that everybody accessing Thinfinity® VirtualUI™ home page will see this profile. Checking this option will disable the Add and Remove buttons.

- **Add**: Press 'Add' to access the windows dialog for selecting Active Directory users.

- **Remove**: Press 'Remove' to remove a user for this application profile.

If you want a user or a user group to access more than one application, you need to create more profiles and then add this user to each profile. The authenticated user will be able to choose from the Web interface which application s/he will connect to.

**Read More:**
- [The 'Licenses' tab](#)
8.1.3 Licenses

In the Thinfinity® VirtualUI™ Manager 'Licenses' tab you will find the following options:

This tab shows the licenses you have currently installed. If you don't have a license yet, you will see a message letting you know how many evaluation days you have left until the trial finishes.

Contact us regarding pricing and/or licensing questions.

Read More:
- The Production Server
Thinfinity® VirtualUI™ Server Manager is a tool to administrate the Thinfinity® VirtualUI™ Server. From its interface you can manage applications profiles, permissions and other settings related to Thinfinity® VirtualUI™ Server. When in development mode, a very similar tool called Development Server Manager is used.

To access Thinfinity® VirtualUI™ Server Manager go to the Start Menu and look for the 'Thinfinity VirtualUI™ Server Manager' shortcut.

Its main menu has two sub-menus:

**File Menu:**
The File Menu is composed of the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save</td>
<td>Click to save any change done on the system Settings.</td>
</tr>
<tr>
<td>Exit</td>
<td>Click on this option to exit Thinfinity® VirtualUI™ Server Manager.</td>
</tr>
</tbody>
</table>

Help Menu:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>Takes you to the application online guide.</td>
</tr>
<tr>
<td>Buy</td>
<td>Takes you to the Cybele Software Buy page.</td>
</tr>
<tr>
<td>About Thinfinity VirtualUI</td>
<td>Click on the About to see the application version and build number.</td>
</tr>
</tbody>
</table>

Read More:
- The 'General' Tab
- The 'Gateways' tab
- The 'RDS' tab
- The 'Applications' tab
- The 'Licences' tab
8.2.1 General

This is how the Thinfinity VirtualUI Server Manager looks in a Standard Mode installation. You will find the following options:

- **Bind to IP**: Use this option to restrict access to the service to one specific IP address. The 'All unassigned' option allows access through all the available IP addresses.

- **Protocol**: Choose between the HTTP and HTTPS protocol.

- **Port**: 6580

- **Communications Settings**
  - **Bind to IP**: (All unassigned)
  - **Protocol**: HTTPS
  - **Port**: 6580

Registered on http://192.168.0.228:6580/

<table>
<thead>
<tr>
<th><strong>Bind to IP</strong></th>
<th>Use this option to restrict access to the service to one specific IP address. The 'All unassigned' option allows access through all the available IP addresses.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Protocol</strong></td>
<td>Choose between the HTTP and HTTPS protocol.</td>
</tr>
<tr>
<td>![Warning Icon]</td>
<td>Press this button to configure HTTP error responses.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>This button is only visible when the protocol is set to HTTPS. Press this button to access the options for replacing the default Thinfinity VirtualUI installed</td>
</tr>
</tbody>
</table>
certificate with your own. Read more about managing the SSL certificates.

This button is only visible when the protocol is set to HTTPS. Press this button to access the HTTPS Security Settings.

<table>
<thead>
<tr>
<th>Port</th>
<th>Choose which port will Thinfinity® VirtualUI™ Server be listening on. If the port is not available, you will see an error message on the status bar.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Log</td>
<td>Press to open the file with the Thinfinity VirtualUI log.</td>
</tr>
</tbody>
</table>

Always remember to press 'Apply' in order to save the changes.

**Read more:**
- Configure HTTP Error Responses
- Managing the SSL Certificate
- HTTPS Security Settings
- The 'Gateways' tab
8.2.1.1 Configure HTTP Error Responses

You can access configuration for the HTTP Error response pages by pressing this button:

which you will find in the Server Manager General tab, when the protocol is set to HTTPS.

You will be presented with the following dialog:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Path</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>401.html</td>
<td>Send File</td>
</tr>
<tr>
<td>402</td>
<td>402.html</td>
<td>Send File</td>
</tr>
<tr>
<td>403</td>
<td>403.html</td>
<td>Send File</td>
</tr>
<tr>
<td>404</td>
<td>404.html</td>
<td>Send File</td>
</tr>
<tr>
<td>409</td>
<td>409.html</td>
<td>Send File</td>
</tr>
<tr>
<td>500</td>
<td>500.html</td>
<td>Send File</td>
</tr>
</tbody>
</table>

This numeric code indicates the status of the response when a browser tries to access content in Thinfinity VirtualUI. The error responses may be displayed in the client browser.

The HTTP status code may indicate whether a request is successful or unsuccessful, and may also reveal the exact reason that a request is unsuccessful.

Path

Shows the path to the error file that will show in Thinfinity VirtualUI.
case of a particular status code. The default path is the 'web' directory in the Thinfinity Virtual installation directory.

| Type          | Shows the Thinfinity VirtualUI action in the event of an error status code:  
|              | - Send file: Thinfinity VirtualUI will show an error page located physically in the server's computer.  
|              | - Redirect: Thinfinity VirtualUI will redirect the page to any web page indicated in the configuration. |
| Add          | Press this button to add a new Custom Error page. Read more about this below. |
| Edit         | Press this button to edit an existing Custom Error Page. Read more about this below. |
| Remove       | Press this button to remove a selected Custom Error Page. |

If you choose to add or edit a Custom Error Page, you will be presented with the following dialog:

![Edit Custom Error Page dialog]

Status Code
Enter the Status Code number that you want to configure.
<table>
<thead>
<tr>
<th><strong>Response Action</strong></th>
<th>Choose whether Thinfinity VirtualUI will show a page that is stored locally or will redirect the user to another web page.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insert Content from file into the error response</strong></td>
<td>Choose this option if you want Thinfinity VirtualUI to show a static page locally stored in your Thinfinity VirtualUI server. Complete the file path by selecting the file you want to show with the button.</td>
</tr>
<tr>
<td><strong>Response with a 302 redirect</strong></td>
<td>Choose this option if you want Thinfinity VirtualUI to redirect users to a web page. Type the Absolute URL to this web page in the field below</td>
</tr>
</tbody>
</table>

Press OK to save the changes.

**Read more:**
- Managing the SSL Certificate
- HTTPS Security Settings
- The Gateways tab

### 8.2.1.2 Managing the SSL Certificate

You can access configuration for managing the SSL certificates by pressing this button:

![SSL Certificate Configuration Button](image)

which you will find in the [Server Manager General tab](#), when the protocol is set to HTTPS.

You will be presented with the following dialog:
An SSL certificate is an effective way to secure a website against unauthorized interception of data. At its simplest, an SSL Certificate is used to identify the website and encrypt all data flowing to and from the Certificate holder's web site. This makes all exchanges between the site and its visitors 100 percent private.

A valid SSL certificate is included with the Thinfinity® VirtualUI™ Server installation and all communications are already encrypted with the product's default certificate. You may want to create your own certificate to identify your company better.

**Managing the SSL Certificate**

1. There are two ways of creating your own SSL certificate:
   
a. Create a **self-signed certificate**
b. Use a **CA Certificate**

2. Once you already have your certificate files, go to the Thinfinity® VirtualUI™ Server Settings 'General' tab.

3. Click on the 'Manage Certificate' option.

4. On this screen you should inform the location of the certificate files, as follows:

**Certificate File**
Inform the path to the certificate file.

**CA File**
If the certificate is issued by a unknown CA, you should fill in the pathname to the CA certificate.

**Private Key**
You should inform the pathname to the certificate private key file.

**Pass Phrase**
Inform the password that was used, if any, when the private key was generated.

Note: The path names can be absolute (C:\MyCertPath\UserThisCert.pem) or relative to
the path where Thinfinity® VirtualUI™ Server is installed (\cert\UserThisCert.perm).

Read more:
- The Default Embedded Certificate
- A Self-signed Certificate
- A CA Certificate
- HTTPS Security Settings
- The Gateways tab
8.2.1.2.1 The Default Embedded Certificate

A certificate called "self-signed.pem" is included with the Thinfinity® VirtualUI™ Server installation. You will find it inside the \cert directory, located inside the Thinfinity® VirtualUI™ Server application path.

If you want to use this default certificate you should have the files set as the image below:

![Manage SSL Certificate](image)

You'll find these settings inside the Thinfinity® VirtualUI™ Server Settings 'General' tab, by clicking on the 'Manage certificate' button.

⚠️ Because this certificate is not issued by a known Certificate Authority (CA), the web browsers will warn you they can not verify its authority.

Read more:
- A Self-signed Certificate
- A CA Certificate
- HTTPS Security Settings
- The Gateways tab
8.2.1.2.2 A Self-signed Certificate

This option is used to create your own self-sign certificate.

1. Go to the Thinfinity® VirtualUI™ Server Settings 'Security' tab.
2. Press the 'Manage certificate' button.
3. Press the 'Create a self-signed certificate' button.
4. Fill in the form below with your organization data:

```
Create self-signed certificate and private key

Country Code:
State:       
Locality:    
Organization:
Organizational Unit:  
Common Name:    
E-Mail address: 
Bits: >= 512

Certificate and private key are written to the same file.
Private key will not be password protected.
```

<table>
<thead>
<tr>
<th><strong>Country Code</strong></th>
<th>The two letter country code of the International Organization for Standardization (ISO 3166)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State</strong></td>
<td>Full unabbreviated name of the state or province your organization is located.</td>
</tr>
<tr>
<td><strong>Locality</strong></td>
<td>Full unabbreviated name of the city where your organization is located.</td>
</tr>
<tr>
<td><strong>Organization</strong></td>
<td>The name your company is legally registered under.</td>
</tr>
</tbody>
</table>
Organizational Unit | Use this field to differentiate between divisions within an organization.
---|---
Common Name | The domain name or URL you plan to use this certificate with.
E-Mail Address | Company e-mail address.
Bits | We recommend using a 2048 length key.

5. The 'Common Name' field should be filled with the server+domain that will be used to access Thinfinity® VirtualUI™ Server (ThinfinityVirtualUI.mycompany.com).

6. Press 'Create'.

7. Select the location where you want the certificate to be stored.

8. The application will start using this self-signed certificate created by you.

⚠️ Because this certificate is not issued by a known Certificate Authority (CA), the web browsers will warn you they can not verify its authority.

Read more:
- A CA Certificate
- HTTPS Security Settings
- The Gateways tab
8.2.1.2.3 A CA Certificate

In order to use this option you will have to get a certificate from a known Certificate Authority (CA). Some CA examples are GoDaddy, VeriSign, Thawte, GeoTrust and Network Solutions.

The CA will ask you for a "certificate request". Create one following the next steps:

1. Go to the Thinfinity® VirtualUI™ Server Settings 'Security' tab.
2. Press the 'Manage certificate' button.
3. Click on the 'Create a certificate request' button.
4. Fill in the form below with your organization data:

<table>
<thead>
<tr>
<th>Certificate Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Country Code:</strong></td>
</tr>
<tr>
<td><strong>State:</strong></td>
</tr>
<tr>
<td><strong>Locality:</strong></td>
</tr>
<tr>
<td><strong>Organization:</strong></td>
</tr>
<tr>
<td><strong>Organizational Unit:</strong></td>
</tr>
<tr>
<td><strong>Common Name:</strong></td>
</tr>
<tr>
<td><strong>E-Mail address:</strong></td>
</tr>
<tr>
<td><strong>Bits:</strong> &gt;= 512</td>
</tr>
</tbody>
</table>

Request and private key are written to different files. Private key will not be password protected.

<table>
<thead>
<tr>
<th><strong>Country Code</strong></th>
<th>The two letter country code of the International Organization for Standardization (ISO 3166)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State</strong></td>
<td>Full unabbreviated name of the state or province your organization is located.</td>
</tr>
</tbody>
</table>
5. The 'Common Name' field should be filled with the server+domain that will be used to access Thinfinity® VirtualUI™ server (ThinfinityVirtualUI.mycompany.com)

6. Press 'Create' and the application will generate two files.

7. The first window will ask you for a location to keep the private key file: "Where do you want the private key file to be stored".
   a. Inform a name for your private key.
   b. Select a place to keep it safe.
   c. Press the 'Save' button.

8. The second window will ask you for a location to keep the request file: "Where do you want the request file to be stored.".
   a. Inform a name for the request file.
   b. Select a directory where you can find the file later on to send to the CA.
   c. Press the 'Save' button.

9. The first file is the certificate private key. It should always be kept safe with you.

10. Send only the request file to the CA.

After the CA validation process, place the certificate they sent to you in the Thinfinity® VirtualUI™ Server cert directory and inform the path to the files on Thinfinity® VirtualUI™ Server Manager, Manage Certificate option (Certificate file, CA file and Private Key).

Read more:
- HTTPS Security Settings
- The Gateways tab
8.2.1.3 HTTPS Security Settings

You can access the HTTPS Security Settings by pressing this button:

![Lock Icon]

which you will find in the Server Manager General tab, when the protocol is set to HTTPS.

You will be presented with the following dialog:

- **Encryption Methods**
  - Select the HTTPS encryption methods you want Thinfinity VirtualUI to support.

- **Default**
  - Select which of the supported HTTPS encryption method is the default. When a connection is made with a browser that doesn't support the default encryption method, Thinfinity VirtualUI will negotiate the security with other supported encryption methods on this list.

**Read more:**
- [The Gateways tab]

8.2.2 Gateways

In the Thinfinity® VirtualUI™ Manager 'Gateways' tab you will find the following options:
**Network ID**

The network ID identifies this installation. Thinfinity VirtualUI Servers that want to share their resources through one or more Gateways must match their Network ID.

Press this button to see and/or change the Network ID. The default value is a random string but you can change it to something more descriptive.

**Gateway List**

A list of the gateways that a user can connect to in order to access this server's resources.

For a typical installation, with no load balancing architecture, leave it blank.

**Add**

Add a new gateway to the Gateway List. Only if you will use **Scaling and Load Balancing**.

**Remove**

Remove a selected gateway from the Gateway List.
Check out the Scaling and Load Balancing section to learn about more options.

Read more:
- The 'RDS' tab
- The 'Applications' tab
- The 'Licenses' tab
- Scaling and Load Balancing

8.2.3 RDS

In the Thinfinity® VirtualUI™ manager 'RDS' tab you will find the following options:

Standard Mode installation:

Thinfinity® VirtualUI™ makes use of an interactive session. The default setting of Thinfinity® VirtualUI™ is to run applications under the console session, but it can be configured to do it under a Remote Desktop Services session. Only one Windows session will be required for all the connections.

For the production environment, it is recommended to set Thinfinity® VirtualUI™ to run applications under its own Remote Desktop Services session. This will ensure that the service is available at all times. Alternatively, you can choose to have Thinfinity® VirtualUI™ run the applications under the console session by configuring the Auto Logon feature on your Windows operating system.

- Use this account for the Remote Desktop Services session:
  - Username:
  - Password:

Test

Show Log

Apply

Close
Use this account for the Remote Desktop Services session:  

Check this option to enable Thinfinity VirtualUI to run applications under a separate Remote Desktop Services session.

User name  
Enter the username for the Remote Desktop Services session you want Thinfinity VirtualUI to run applications under.

Password  
Enter the password for the Remote Desktop Services session you want Thinfinity VirtualUI to run applications under.

Test  
Test the credentials entered to verify that the username and password are correct and can access RDS.

Load Balancing Mode installation:

Thinfinity® VirtualUI™ makes use of an interactive session. The default setting of Thinfinity® VirtualUI™ is to run applications under the console session, but it can be configured to do it under a Remote Desktop Services session. Only one Windows session will be required for all the connections.

For the production environment, it is recommended to set Thinfinity® VirtualUI™ to run applications under its own Remote Desktop Services session. This will ensure that the service is available at all times. Alternatively, you can choose to have Thinfinity® VirtualUI™ run the applications under the console session by configuring the Auto Login feature on your Windows operating system.
This table shows the RDS sessions that will handle Thinfinity VirtualUI connections. You can use more than one session when you have a lot of connections, so that it is distributed between different sessions in the same computer. You can combine this with a [Scaling and Load Balancing](#).

<table>
<thead>
<tr>
<th>User name</th>
<th>This table shows the RDS sessions that will handle Thinfinity VirtualUI connections. You can use more than one session when you have a lot of connections, so that it is distributed between different sessions in the same computer. You can combine this with a <a href="#">Scaling and Load Balancing</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Press this button to add another RDS account.</td>
</tr>
<tr>
<td>Remove</td>
<td>Press this button to remove a selected RDS account.</td>
</tr>
</tbody>
</table>

Always remember to press 'Apply' in order to save the changes.

**Read More:**
- [The 'Applications' tab](#)
- [The 'Licenses' tab](#)
- [Scaling and Load Balancing](#)
8.2.4 Applications

The 'Applications' tab will allow you to configure the applications' locations and settings as well as the user permissions to access them.

- **Application List:**
  - **Name**
  - **Target**

<table>
<thead>
<tr>
<th>Name</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>RibbonDemo</td>
<td>C:\My Demo\RibbonDemo\RibbonDemo.exe</td>
</tr>
<tr>
<td>Thinfinity Virtual</td>
<td><a href="http://www.cybele.com/helps/thinfinity/virtualui">http://www.cybele.com/helps/thinfinity/virtualui</a></td>
</tr>
</tbody>
</table>

  - **Allowed users and groups for selected application:**
    - **Anonymous access**

  - **Database path:** C:\ProgramData\Cybele Software\Thinfinity\VirtualUI\DB

- **Application List:**
  - **Name:** Shows the name of the application.
  - **Target:** Shows the application path or the web address for Web Link profiles.
Add
Press this button to add a new application.

Edit
Select an application and press this button to edit it.

Remove
Select an application and press this button to remove it.

Allowed users and groups for selected profile
See here the allowed users or group(s) of users for the selected application. If you want to change the permissions, edit the application.

Database path
Path to the profile database.

Always remember to press 'Apply' in order to save the changes.

Read More:
- Application Profile
- Weblink Profile
- The 'Licenses' Tab
- Scaling and Load Balancing
8.2.4.1 Application Profile

When you edit or add an application profile you will be presented with this screen below. The radio button 'Application' must be checked.

These are the profile properties you can edit:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Use this field to change the application name.</td>
</tr>
<tr>
<td>Virtual Path</td>
<td>The Virtual Path will create a unique URL address for this connection. The complete path will consist of: http(s)://ip:port/VirtualPath/. The users can then create a web shortcut to this connection in particular and bypass the Thinfinity VirtualUI web interface.</td>
</tr>
<tr>
<td>Home Page</td>
<td>Choose the landing HTML page for the application.</td>
</tr>
<tr>
<td><strong>Open</strong></td>
<td>Press this button to look for the Home Page.</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td><strong>Access Key</strong></td>
<td>This is a unique key for this application profile. The value is used to identify the application when implementing access through the <a href="#">One-Time-URL</a>.</td>
</tr>
<tr>
<td><strong>Icon</strong></td>
<td>Click on the Icon gray box to load an image to be associated with the profile. The image will be presented along with the profile name on the web interface profiles selection.</td>
</tr>
<tr>
<td><strong>Application/Web link</strong></td>
<td>Select the Application option to have a regular profile that gives access to an application. If you select the Web link radio button, this profile will behave like a Web Hyperlink.</td>
</tr>
<tr>
<td><strong>Default Application</strong></td>
<td>Check this option to make this profile the default application: the authenticated user will connect to this profile directly instead of choosing between the available profiles. The rest of the profiles can be accessed by their Virtual Path.</td>
</tr>
</tbody>
</table>

The properties located inside the tabs will be described throughout the next subtopics.

**Read More:**
- The 'General' Tab
- The 'Credentials' Tab
- The 'Permissions' Tab
- Weblink Profile
8.2.4.1.1 General

In the Application Profile Editor 'General' tab you will find the following options:

<table>
<thead>
<tr>
<th>General</th>
<th>Credentials</th>
<th>Permissions</th>
</tr>
</thead>
</table>

- **Program path and file name**: Specify the complete path that gives access to the application executable file.
- **Arguments**: Application arguments.
- **Start in the following folder**: Inform a context directory for the application set on the 'Program path and file name' field.
- **Resolution**: Choose from the available list of resolutions including 'Fit to browser window' and 'Fit to screen', ideal for hiding the browser and working on a full screen mode.
- **Idle Timeout**: Set a timeout in minutes if you want Thinfinity® VirtualUI™ Server to wait this period before killing the application once the browser has been closed. Timeout 0 will kill the application immediately after the browser has been closed.

**Read More:**
- The 'Credentials' Tab
- The 'Permissions' Tab
8.2.4.1.2 Credentials

In the Thinfinity® VirtualUI™ Application Editor 'Credentials' tab, you should inform the mode for logging into the specified application:

- **Use the authenticated credentials**
  
  Use the same credentials entered in the browser for Thinfinity VirtualUI (specified in the 'Permissions' tab). Note: If the credentials are correct, this option will connect the user automatically when selecting the application, or after authenticating for Thinfinity VirtualUI if this is the only profile for their credentials.

- **Use these credentials**
  
  Complete the credentials used to access the computer. Note: If the credentials are correct, this option will connect the user automatically when selecting the application, or after authenticating for Thinfinity VirtualUI if this is the only profile for their credentials.

**Read More:**
- [The 'Permissions' Tab](#)
8.2.4.1.3 Permissions

Select the users that will access this application. If you don't select any user, this application will not be accessed.

These are the options you will find on the Application Profile Editor 'Permissions' tab:

- **Allow anonymous access**
  - Check this option to make this application available without any authentication. Use this option if you want this profile to be available for everyone. This means that everybody accessing Thinfinity VirtualUI will have access to this application. Checking this option will disable the Add and Remove buttons.

- **Add**
  - Press 'Add' to access the windows dialog for selecting Active Directory users.

- **Remove**
  - Press 'Remove' to remove a user for this profile.

If you want a user or a user group to access more than one application, you need to create more application profiles and then add this user to each profile. The authenticated user will be able to choose from the available application profiles on the Web interface.

**Read More:**
- [Weblink Profile](#)
8.2.4.2 Weblink Profile

When you edit or add a Web Link profile you will be presented with the screen below. The 'Web Link' radio button must be marked.

These are the profile properties you can edit:

<table>
<thead>
<tr>
<th>Name</th>
<th>Use this field to change the profile name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Path</td>
<td>The Virtual Path will create a unique URL address for this connection. The complete path will consist of: http(s)://ThinfinityVirtualUIDomain:port/VirtualPath/. The users can then create a web shortcut to this connection in particular and bypass the Thinfinity VirtualUI web interface.</td>
</tr>
<tr>
<td>Access Key</td>
<td>This is a unique key for this application profile. The value is used to identify the application when implementing access through the One-Time-URL method.</td>
</tr>
<tr>
<td>Icon</td>
<td>Click on the Icon gray box to load an image to be associated with the application. The image will be presented along with the application name on the web interface.</td>
</tr>
<tr>
<td>Web link / Application profile</td>
<td>Select the Weblink option to have a profile that connects to a Web link. These links will be shown along with all the other applications on the Thinfinity VirtualUI start page.</td>
</tr>
<tr>
<td>Default Application</td>
<td>Check this option to make this profile the default application: the authenticated user will connect to this profile directly instead of choosing between the available profiles. The rest of the profiles can be accessed by their Virtual Path.</td>
</tr>
<tr>
<td>Web URL</td>
<td>Inform in this field the URL that you want this application profile to connect to.</td>
</tr>
<tr>
<td>Get Icon</td>
<td>Retrieve an icon from the specified Web URL to be used in the web link profile</td>
</tr>
</tbody>
</table>

The properties located inside the other tabs will be described throughout the next subtopics.

Read more:
- The 'Permissions' tab
### 8.2.4.2.1 Permissions

Select the users that will access this application profile. If you don't select any users, this profile will not be available from the Web interface. These are the options you will find under the 'Permissions' tab:

<table>
<thead>
<tr>
<th>Web Link</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ Allow anonymous access</td>
<td></td>
</tr>
</tbody>
</table>

**Group or user names:**

<table>
<thead>
<tr>
<th>Allow anonymous access</th>
<th>Check this option to make this application available without any authentication. Use this option, if you want this profile to be available for everyone. This means that everybody accessing Thinfinity® VirtualUI™ home page will see this profile. Checking this option will disable the Add and Remove buttons.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Press 'Add' to access the windows dialog for selecting Active Directory users.</td>
</tr>
<tr>
<td>Remove</td>
<td>Press 'Remove' to remove a user for this application profile.</td>
</tr>
</tbody>
</table>

If you want a user or a user group to access more than one application, you need to create more profiles and then add this user to each profile. The authenticated user will be able to choose from the Web interface which application s/he will connect to.

**Read More:**
- The 'Licenses' tab
In the Thinfinity® VirtualUI™ Manager 'Licenses' tab you will find the following options:

This tab shows the licenses you have currently installed. If you don't have a license yet, you will see a message letting you know how many evaluation days you have left until the trial finishes.

Press 'Register' to add your license information.

Contact us regarding pricing and/or licensing questions.
8.3 Gateway

The Gateway Manager is a tool to configure gateway options in a [Load Balancing](#) scenario.

Install Thinfinity VirtualUI as a Gateway Role and look for the 'Thinfinity VirtualUI Gateway' shortcut in the Start Menu.

![Gateway Manager](image)

Its main menu has two sub-menus:

**File Menu:**

- Save
- Close

The File Menu is composed of the following options:
Save  
Click to save any change done on the system Settings.

Close  
Click on this option to exit Thinfinity VirtualUI Gateway Manager.

Help Menu:

Help
About Thinfinity VirtualUI...

The Help Menu is composed of the following options:

About Thinfinity VirtualUI  
Click on the About to see the application version and build number.

The General tab presents the following options:

Bind to IP  
Use this option to restrict access to the service to one specific IP address. The 'All unassigned' option allows access through all the available IP addresses.

Protocol  
Choose between the http and https protocol.

Configure HTTP Error responses.

This button is only visible when the protocol is set to HTTPS. Press this button to access the options for replacing the default Thinfinity VirtualUI installed certificate with your own. Read more about managing the SSL certificates.

Port  
Choose which port will Thinfinity® VirtualUI™ Gateway be listening on. If the port is not available, you will see an error message on the status bar.

Network ID  
The network ID identifies this gateway services installation. Thinfinity VirtualUI Servers that want to share their resources through this this gateway must match this Network ID.

Press this button to see and/or change the Network ID. The default value is a random string.
but you can change it to something more descriptive.

| Show Log | Press to open the file with the Thinfinity VirtualUI log. |

9 Symbol Reference

ActiveX Interfaces

.NET Classes

Delphi Classes

C++ Classes

VirtualUI.sdk.min.js

9.1 ActiveX Interfaces

<table>
<thead>
<tr>
<th>Libraries</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>VirtualUI</td>
<td>Main class. Has methods, properties and events to allow the activation and control the behavior of VirtualUI.</td>
</tr>
<tr>
<td>VirtualUIS</td>
<td></td>
</tr>
</tbody>
</table>

9.1.1 VirtualUI Runtime Library

Main class. Has methods, properties and events to allow the activation and control the behavior of VirtualUI.

<table>
<thead>
<tr>
<th>Enumerations</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><code>IJSDataType</code></td>
<td>This is record VirtualUI::IJSDataType.</td>
</tr>
<tr>
<td><code>Orientation</code></td>
<td>This is record VirtualUI::Orientation.</td>
</tr>
<tr>
<td><code>MouseMoveGestureStyle</code></td>
<td>This is record VirtualUI::MouseMoveGestureStyle.</td>
</tr>
<tr>
<td><code>MouseMoveGestureAction</code></td>
<td>This is record VirtualUI::MouseMoveGestureAction.</td>
</tr>
</tbody>
</table>
## Group

**ActiveX Interfaces**

## Interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVirtualUI</td>
<td>Main interface. Has methods, properties and events to allow the activation and control the behavior of VirtualUI.</td>
</tr>
<tr>
<td>IBrowserInfo</td>
<td>Contains information regarding the end-user's screen, web browser, the window containing VirtualUI Viewer and VirtualUI Viewer itself. The VirtualUI Viewer runs inside an HTML DIV element contained in a frame of browser window on the end-user's application page.</td>
</tr>
<tr>
<td>IDevServer</td>
<td>Contains properties to manage the VirtualUI Development Server as well as the access from the developer's web browser.</td>
</tr>
<tr>
<td>IClientSettings</td>
<td>Contains settings that are synchronized with the VirtualUI Viewer.</td>
</tr>
<tr>
<td>IJSValue</td>
<td>Base interface for IJSNamedValue, IJSProperty and IJSArgument.</td>
</tr>
<tr>
<td>IJSNamedValue</td>
<td>Base interface for IJSProperty and IJSArgument.</td>
</tr>
<tr>
<td>IJSObject</td>
<td>Represents a custom remotable object.</td>
</tr>
<tr>
<td>IJSObjects</td>
<td>Collection of IJSObjects</td>
</tr>
<tr>
<td>IJSProperty</td>
<td>Represents a property in a custom remotable object.</td>
</tr>
<tr>
<td>IJSProperties</td>
<td>Collection of properties.</td>
</tr>
<tr>
<td>IJSArgument</td>
<td>Represents a parameter in events and methods.</td>
</tr>
<tr>
<td>IJSArguments</td>
<td>Collection of IJSArgument.</td>
</tr>
<tr>
<td>IJSMethod</td>
<td>This interface represents an method called on the client side and executed on the server side.</td>
</tr>
<tr>
<td>IJSMethods</td>
<td>Collection of methods.</td>
</tr>
<tr>
<td>IJSEvent</td>
<td>This interface represents an event fired on the server side and raised on the client side.</td>
</tr>
<tr>
<td>IJSEvents</td>
<td>Collection of events.</td>
</tr>
<tr>
<td>IJSBinding</td>
<td>Used as a callback interface for binding external data source with the IJSProperty data.</td>
</tr>
<tr>
<td>IJSCallback</td>
<td>Used as a callback interface for executing the remote calling to IJSMethod.</td>
</tr>
<tr>
<td>IEvents</td>
<td>General VirtualUI events.</td>
</tr>
<tr>
<td>IJSObjectEvents</td>
<td>This is class VirtualUI::IJSObjectEvents.</td>
</tr>
</tbody>
</table>
9.1.1.1 VirtualUI::IJSDataType Enumeration

IDL
[ uuid(B81C7C6D-CDD4-4268-A769-C46EC04FD58B) ]
enum IJSDataType {
    JSDT_NULL = 0,
    JSDT_STRING = 1,
    JSDT_INT = 2,
    JSDT_BOOL = 3,
    JSDT_FLOAT = 4,
    JSDT_JSON = 5
};

File
VirtualUIX.ridl

Description
This is record VirtualUI::IJSDataType.

Library
VirtualUI Runtime Library

9.1.1.2 VirtualUI::Orientation Enumeration

IDL
[ uuid(0715C547-D231-4F69-9D25-44164C90DA88) ]
enum Orientation {
    PORTRAIT = 0,
    LANDSCAPE = 1
};

File
VirtualUIX.ridl

Description
This is record VirtualUI::Orientation.

Library
VirtualUI Runtime Library

9.1.1.3 VirtualUI::MouseMoveGestureStyle Enumeration

IDL
[ uuid(B1A699F2-F16C-49AA-93DD-1A056936AA87) ]
enum MouseMoveGestureStyle {
File
VirtualUIX.ridl

Description
This is record VirtualUI::MouseMoveGestureStyle.

Library
VirtualUI Runtime Library

9.1.1.4 VirtualUI::MouseMoveGestureAction Enumeration

IDL
[ uuid(EEC4FF42-7574-4250-8433-E14593AB7E7B) ]
enum MouseMoveGestureAction {
    MM_ACTION_MOVE = 0,
    MM_ACTION_WHEEL = 1
};

File
VirtualUIX.ridl

Description
This is record VirtualUI::MouseMoveGestureAction.

Library
VirtualUI Runtime Library

9.1.1.5 IVirtualUI Interface

Main interface. Has methods, properties and events to allow the activation and control the behavior of VirtualUI.

Class Hierarchy

IDL
[ uuid(4B85F81B-72A2-4FCD-9A6B-9CAC24B7A511), helpstring("Interface for VirtualUI Object"), dual, oleautomation ]
interface IVirtualUI : IDispatch;

File
Library

VirtualUI Runtime Library

9.1.1.5.1 IVirtualUI Methods

The methods of the IVirtualUI class are listed here.

Interface

IVirtualUI Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DevMode</td>
<td>Gets/sets the development mode.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Enables/disables VirtualUI for the container application.</td>
</tr>
<tr>
<td>StdDialogs</td>
<td>Enables/disables the use of standard dialogs.</td>
</tr>
<tr>
<td>Start</td>
<td>Starts the VirtualUI's activation process. Returns true if VirtualUI was fully activated or false if the passed timeout expired.</td>
</tr>
<tr>
<td>Stop</td>
<td>Deactivates VirtualUI, closing the connection with the end-user's web browser</td>
</tr>
<tr>
<td>Active</td>
<td>Returns the VirtualUI's state.</td>
</tr>
<tr>
<td>BrowserInfo</td>
<td>Contains information regarding the end-user's environment.</td>
</tr>
<tr>
<td>DevServer</td>
<td>Allows for managing the Development Server.</td>
</tr>
<tr>
<td>ClientSettings</td>
<td>Controls some working parameters on the client side.</td>
</tr>
<tr>
<td>DownloadFile</td>
<td>Sends the specified file to the end-user's web browser for saving it in the remote machine.</td>
</tr>
<tr>
<td>PrintPdf</td>
<td>Sends the specified PDF file to be shown on the end-user's web browser.</td>
</tr>
<tr>
<td>OpenLinkDlg</td>
<td>This is OpenLinkDlg, a member of class IVirtualUI.</td>
</tr>
<tr>
<td>SendMessage</td>
<td>Sends a data string to the web browser.</td>
</tr>
<tr>
<td>AllowExecute</td>
<td>Allows the execution of the passed application.</td>
</tr>
<tr>
<td>SetImageQualityByWnd</td>
<td>This is SetImageQualityByWnd, a member of class IVirtualUI.</td>
</tr>
<tr>
<td>UploadFile</td>
<td>This is UploadFile, a member of class IVirtualUI.</td>
</tr>
<tr>
<td>TakeScreenshot</td>
<td>This is TakeScreenshot, a member of class IVirtualUI.</td>
</tr>
</tbody>
</table>
9.1.5.1.1 DevMode Method

Gets/sets the development mode.

Remarks
When in development mode, applications executed under the IDE, connect to the Development Server, allowing the access to the application from the browser while in debugging.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVirtualUI::DevMode ([in] VARIANT_BOOL)</td>
<td>This is DevMode, a member of class IVirtualUI.</td>
</tr>
<tr>
<td>IVirtualUI::DevMode ([out, retval] VARIANT_BOOL*)</td>
<td>This is DevMode, a member of class IVirtualUI.</td>
</tr>
</tbody>
</table>

Group
IVirtualUI Methods

IDL
[propget, id(0x00000069)]
HRESULT _stdcall DevMode([out, retval] VARIANT_BOOL* Value);

Description
This is DevMode, a member of class IVirtualUI.

Group
DevMode Method

IDL
[propput, id(0x00000069)]
HRESULT _stdcall DevMode([in] VARIANT_BOOL Value);

Description
This is DevMode, a member of class IVirtualUI.

Group
DevMode Method
9.1.1.5.1.2 Enabled Method

Enables/disables VirtualUI for the container application.

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVirtualUI::Enabled ([in] VARIANT_BOOL)</td>
<td>This is Enabled, a member of class IVirtualUI.</td>
</tr>
<tr>
<td>IVirtualUI::Enabled ([out, retval] VARIANT_BOOL*)</td>
<td>This is Enabled, a member of class IVirtualUI.</td>
</tr>
</tbody>
</table>

### Group

IVirtualUI Methods

### IDL

[propget, id(0x0000006A)]

HRESULT _stdcall Enabled([out, retval] VARIANT_BOOL* Value);

### Description

This is Enabled, a member of class IVirtualUI.

### Group

Enabled Method

### IDL

[propput, id(0x0000006A)]

HRESULT _stdcall Enabled([in] VARIANT_BOOL Value);

### Description

This is Enabled, a member of class IVirtualUI.

### Group

Enabled Method

9.1.1.5.1.3 StdDialogs Method

Enables/disables the use of standard dialogs.

### Remarks

When set to false, the standard save, open and print dialogs are replaced by native browser ones, enabling you to extend the operations to the remote computer.
Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVirtualUI::StdDialogs ([in] VARIANT_BOOL)</td>
<td>This is StdDialogs, a member of class IVirtualUI.</td>
</tr>
<tr>
<td>IVirtualUI::StdDialogs ([out, retval] VARIANT_BOOL*)</td>
<td>This is StdDialogs, a member of class IVirtualUI.</td>
</tr>
</tbody>
</table>

Group

IVirtualUI Method

**IDL**

[propget, id(0x0000006B)]

HRESULT _stdcall StdDialogs([out, retval] VARIANT_BOOL* Value);

**Description**

This is StdDialogs, a member of class IVirtualUI.

Group

StdDialogs Method

**IDL**

[propput, id(0x0000006B)]

HRESULT _stdcall StdDialogs([in] VARIANT_BOOL Value);

**Description**

This is StdDialogs, a member of class IVirtualUI.

9.1.1.5.1.4 IVirtualUI::Start Method

Starts the VirtualUI's activation process. Returns true if VirtualUI was fully activated or false if the passed timeout expired.

**IDL**

[id(0x00000066)]

HRESULT _stdcall Start([in] long Timeout, [out, retval] VARIANT_BOOL* OutRetVal);

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] long Timeout</td>
<td>Maximum time, in seconds, until the activation process is canceled. Defaults to 60 seconds.</td>
</tr>
</tbody>
</table>
Remarks

To fully activate VirtualUI, the connection with the end-user's web browser must be established within the time specified by the Timeout parameter.

Group

IVirtualUI Methods

9.1.1.5.1.5 IVirtualUI::Stop Method

Deactivates VirtualUI, closing the connection with the end-user's web browser.

IDL

[id(0x00000067)]
HRESULT _stdcall Stop();

Group

IVirtualUI Methods

9.1.1.5.1.6 IVirtualUI::Active Method

Returns the VirtualUI's state.

IDL

[propget, id(0x00000068)]
HRESULT _stdcall Active([out, retval] VARIANT_BOOL* Value);

Group

IVirtualUI Methods

9.1.1.5.1.7 IVirtualUI::BrowserInfo Method

Contains information regarding the end-user's environment.

IDL

[propget, id(0x00000065)]
HRESULT _stdcall BrowserInfo([out, retval] IBrowserInfo** Value);

Group

IVirtualUI Methods

9.1.1.5.1.8 IVirtualUI::DevServer Method

Allows for managing the Development Server.

IDL
Group

IVirtualUI Methods

9.1.1.5.1.9 IVirtualUI::ClientSettings Method

Controls some working parameters on the client side.

**IDL**

```idl
[propget, id(0x0000006F)]
HRESULT _stdcall ClientSettings([out, retval] IClientSettings** Value);
```

Group

IVirtualUI Methods

9.1.1.5.1.10 IVirtualUI::DownloadFile Method

Sends the specified file to the end-user's web browser for saving it in the remote machine.

**IDL**

```idl
[id(0x00000070)]
HRESULT _stdcall DownloadFile([in] BSTR LocalFilename, [in] BSTR RemoteFilename, [in] BSTR MimeType);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR LocalFilename</td>
<td>Name of the local file to be sent.</td>
</tr>
<tr>
<td>[in] BSTR RemoteFilename</td>
<td>Name of the file in the remote machine.</td>
</tr>
<tr>
<td>[in] BSTR MimeType</td>
<td>content-type of the file. If specified, the content will be handled by browser. Leave blank to force download.</td>
</tr>
</tbody>
</table>

Group

IVirtualUI Methods

9.1.1.5.1.11 IVirtualUI::PrintPdf Method

Sends the specified PDF file to be shown on the end-user's web browser.

**IDL**

```idl
[id(0x00000071)]
HRESULT _stdcall PrintPdf([in] BSTR AFileName);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFileName</td>
<td>Name</td>
</tr>
</tbody>
</table>
PrintPDF is similar to DownloadFile, except that it downloads the file with a content-type: application/pdf.

**Group**

**IVirtualUI Methods**

### 9.1.1.5.1.12  IVirtualUI::OpenLinkDlg Method

**IDL**

```idl
[id(0x00000072)]
HRESULT _stdcall OpenLinkDlg([in] BSTR url, [in] BSTR caption);
```

**Description**

This is OpenLinkDlg, a member of class IVirtualUI.

**Group**

**IVirtualUI Methods**

### 9.1.1.5.1.13  IVirtualUI::SendMessage Method

Sends a data string to the web browser.

**IDL**

```idl
[id(0x00000073)]
HRESULT _stdcall SendMessage([in] BSTR Data);
```

**Remarks**

This method is used to send custom data to the browser for custom purposes.

**Group**

**IVirtualUI Methods**

### 9.1.1.5.1.14  IVirtualUI::AllowExecute Method

Allows the execution of the passed application.

**IDL**

```idl
[id(0x00000074)]
HRESULT _stdcall AllowExecute([in] BSTR Filename);
```

**Parameters**
## Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR Filename</td>
<td>regular expression specifying the filename(s) of the applications allowed to run.</td>
</tr>
</tbody>
</table>

#### Remarks

Under **VirtualUI** environment, only applications precompiled with **VirtualUI** SDK should be allowed to run. Applications not under **VirtualUI** control, cannot be controlled.

#### Group

**IVirtualUI Methods**

#### 9.1.1.5.1.15 IVirtualUI::SetImageQualityByWnd Method

**IDL**

```
[id(0x00000075)]
HRESULT _stdcall SetImageQualityByWnd([in] long Wnd, [in] BSTR Class, [in] long Quality);
```

**Description**

This is SetImageQualityByWnd, a member of class IVirtualUI.

#### 9.1.1.5.1.16 IVirtualUI::UploadFile Method

**IDL**

```
[id(0x00000076)]
HRESULT _stdcall UploadFile([in] BSTR ServerDirectory);
```

**Description**

This is UploadFile, a member of class IVirtualUI.

#### 9.1.1.5.1.17 IVirtualUI::TakeScreenshot Method

**IDL**

```
[id(0x00000077)]
HRESULT _stdcall TakeScreenshot([in] long Wnd, [in] BSTR FileName, [out, retval] VARIANT_BOOL* OutRetVal);
```

**Description**

This is TakeScreenshot, a member of class IVirtualUI.
Group

IVirtualUI Methods

9.1.1.5.1.18 IVirtualUI::ShowVirtualKeyboard Method

IDL

[id(0x00000078)]
HRESULT _stdcall ShowVirtualKeyboard();

Description

This is ShowVirtualKeyboard, a member of class IVirtualUI.

Group

IVirtualUI Methods

9.1.1.6 IBrowserInfo Interface

Contains information regarding the end-user's screen, web browser, the window containing VirtualUI Viewer and VirtualUI Viewer itself. The VirtualUI Viewer runs inside an HTML DIV element contained in a frame of browser window on the end-user's application page.

Class Hierarchy

IDL

[ uuid(4D9F5347-460B-4275-BDF2-F2738E7F6757), dual, oleautomation ]
interface IBrowserInfo : IDispatch;

File

VirtualUIX.ridl

Library

VirtualUI Runtime Library

9.1.1.6.1 IBrowserInfo Methods

The methods of the IBrowserInfo class are listed here.

Interface

IBrowserInfo Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ViewHeight</td>
<td>Returns the height of the VirtualUI Viewer.</td>
</tr>
<tr>
<td>ViewWidth</td>
<td>Returns the width of the VirtualUI Viewer.</td>
</tr>
<tr>
<td>BrowserWidth</td>
<td>Returns the width of the window containing the VirtualUI Viewer.</td>
</tr>
<tr>
<td>BrowserHeight</td>
<td>Returns the height of the window containing the VirtualUI Viewer.</td>
</tr>
<tr>
<td>ScreenWidth</td>
<td>Returns the width of the end-user's monitor screen.</td>
</tr>
<tr>
<td>ScreenHeight</td>
<td>Returns the height of the end-user's monitor screen.</td>
</tr>
<tr>
<td>Username</td>
<td>Returns the logged-on Username.</td>
</tr>
<tr>
<td>IPAddress</td>
<td>Returns the client's IP address.</td>
</tr>
<tr>
<td>UserAgent</td>
<td>Returns the browser's User Agent string</td>
</tr>
<tr>
<td>ScreenResolution</td>
<td>Returns the application screen resolution defined in the application profile.</td>
</tr>
<tr>
<td>Orientation</td>
<td>Returns the browser's orientation.</td>
</tr>
<tr>
<td>UniqueBrowserId</td>
<td>This is UniqueBrowserId, a member of class IBrowserInfo.</td>
</tr>
<tr>
<td>GetCookie</td>
<td>This is GetCookie, a member of class IBrowserInfo.</td>
</tr>
<tr>
<td>SetCookie</td>
<td>This is SetCookie, a member of class IBrowserInfo.</td>
</tr>
<tr>
<td>Location</td>
<td>This is Location, a member of class IBrowserInfo.</td>
</tr>
</tbody>
</table>

### 9.1.1.6.1.1 ViewHeight Method

Returns the height of the VirtualUI Viewer.

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBrowserInfo::ViewHeight</td>
<td>This is ViewHeight, a member of class IBrowserInfo.</td>
</tr>
<tr>
<td>([in] long)</td>
<td></td>
</tr>
<tr>
<td>IBrowserInfo::ViewHeight</td>
<td>This is ViewHeight, a member of class IBrowserInfo.</td>
</tr>
<tr>
<td>([out, retval] long*)</td>
<td></td>
</tr>
</tbody>
</table>

### Group

**IBrowserInfo Methods**

**IDL**

```c
[propget, id(0x00000066)]
HRESULT _stdcall ViewHeight([out, retval] long* Value);
```

### Description
This is ViewHeight, a member of class IBrowserInfo.

### Group

#### ViewHeight Method

**IDL**

```idl
[propget, id(0x00000065)]
HRESULT _stdcall ViewWidth([out, retval] long *Value);
```

### Description

This is ViewHeight, a member of class IBrowserInfo.

### Group

#### IBrowserInfo Methods

**IDL**

```idl
[propget, id(0x00000065)]
HRESULT _stdcall ViewWidth([out, retval] long *Value);
```

### Description

This is ViewWidth, a member of class IBrowserInfo.

### Group

#### ViewWidth Method

**IDL**

```idl
[propput, id(0x00000066)]
HRESULT _stdcall ViewHeight([in] long Value);
```

### Description

This is ViewHeight, a member of class IBrowserInfo.

### Group

#### ViewWidth Method

**IDL**

```idl
[propput, id(0x00000066)]
HRESULT _stdcall ViewHeight([in] long Value);
```
Description
This is ViewWidth, a member of class IBrowserInfo.

Group
ViewWidth Method

9.1.1.6.1.3 IBrowserInfo::BrowserWidth Method

Returns the width of the window containing the VirtualUI Viewer.

IDL
[propget, id(0x00000067)]
HRESULT _stdcall BrowserWidth([out, retval] long* Value);

Group
IBrowserInfo Methods

9.1.1.6.1.4 IBrowserInfo::BrowserHeight Method

Returns the height of the window containing the VirtualUI Viewer.

IDL
[propget, id(0x00000068)]
HRESULT _stdcall BrowserHeight([out, retval] long* Value);

Group
IBrowserInfo Methods

9.1.1.6.1.5 IBrowserInfo::ScreenWidth Method

Returns the width of the end-user's monitor screen.

IDL
[propget, id(0x00000069)]
HRESULT _stdcall ScreenWidth([out, retval] long* Value);

Group
IBrowserInfo Methods

9.1.1.6.1.6 IBrowserInfo::ScreenHeight Method

Returns the height of the end-user's monitor screen.

IDL
[propget, id(0x0000006A)]
GROUP IBrowserInfo Methods

9.1.1.6.1.7 IBrowserInfo::Username Method

Returns the logged-on Username.

IDL
[propget, id(0x0000006B)]
HRESULT _stdcall Username([out, retval] BSTR* Value);

GROUP IBrowserInfo Methods

9.1.1.6.1.8 IBrowserInfo::IPAddress Method

Returns the client's IP address.

IDL
[propget, id(0x0000006D)]
HRESULT _stdcall IPAddress([out, retval] BSTR* Value);

GROUP IBrowserInfo Methods

9.1.1.6.1.9 IBrowserInfo::UserAgent Method

Returns the browser's User Agent string

IDL
[propget, id(0x0000006E)]
HRESULT _stdcall UserAgent([out, retval] BSTR* Value);

GROUP IBrowserInfo Methods

9.1.1.6.1.10 IBrowserInfo::ScreenResolution Method

Returns the application screen resolution defined in the application profile.

IDL
[propget, id(0x0000006F)]
HRESULT _stdcall ScreenResolution([out, retval] long* Value);
9.1.1.6.1.11 IBrowserInfo::Orientation Method

Returns the browser's orientation.

**IDL**

```idl
[propget, id(0x000000C9)]
HRESULT _stdcall Orientation([out, retval] enum Orientation* Value);
```

**Description**

This is the Orientation method, a member of class IBrowserInfo.

9.1.1.6.1.12 IBrowserInfo::UniqueBrowserId Method

**IDL**

```idl
[propget, id(0x000000CA)]
HRESULT _stdcall UniqueBrowserId([out, retval] BSTR* Value);
```

**Description**

This is UniqueBrowserId, a member of class IBrowserInfo.

9.1.1.6.1.13 IBrowserInfo::GetCookie Method

**IDL**

```idl
[id(0x000000CB)]
HRESULT _stdcall GetCookie([in] BSTR Name, [out, retval] BSTR* Value);
```

**Description**

This is GetCookie, a member of class IBrowserInfo.

9.1.1.6.1.14 IBrowserInfo::SetCookie Method

**IDL**

```idl
[id(0x000000CC)]
HRESULT _stdcall SetCookie([in] BSTR Name, [in] BSTR Value, [in] BSTR Expires);
```

**Description**

This is SetCookie, a member of class IBrowserInfo.
This is SetCookie, a member of class IBrowserInfo.

**Group**

**IBrowserInfo Methods**

### 9.1.1.6.1.15 IBrowserInfo::Location Method

**IDL**

```idl
HRESULT _stdcall Location([out, retval] BSTR* Value);
```

**Description**

This is Location, a member of class IBrowserInfo.

**Group**

**IBrowserInfo Methods**

### 9.1.1.7 IDevServer Interface

Contains properties to manage the VirtualUI Development Server as well as the access from the developer's web browser.

**Class Hierarchy**

```
IDispatch -> VirtualUI::IDevServer
```

**IDL**

```idl
[ uuid(B3EAC0CA-D7AB-4AB1-9E24-84A63C8C3F80), dual, oleautomation ]
interface IDevServer : IDispatch;
```

**File**

VirtualUIX.ridl

**Library**

VirtualUI Runtime Library

### 9.1.1.7.1 IDevServer Methods

The methods of the IDevServer class are listed here.

**Interface**

IDevServer Interface
### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Enables/disables the Development Server.</td>
</tr>
<tr>
<td>Port</td>
<td>Gets/sets the Development Server's TCP/IP listening port.</td>
</tr>
<tr>
<td>StartBrowser</td>
<td>Instructs VirtualUI whether start or not the local web browser upon VirtualUI activation.</td>
</tr>
</tbody>
</table>

#### 9.1.1.7.1.1 Enabled Method

Enables/disables the Development Server.

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDevServer::Enabled ([in] VARIANT_BOOL)</td>
<td>This is Enabled, a member of class IDevServer.</td>
</tr>
<tr>
<td>IDevServer::Enabled ([out, retval] VARIANT_BOOL*)</td>
<td>This is Enabled, a member of class IDevServer.</td>
</tr>
</tbody>
</table>

#### Description

This is Enabled, a member of class IDevServer.

---

© 2015, Cybele Software, Inc.
9.1.1.7.1.2 Port Method

Gets/sets the Development Server's TCP/IP listening port.

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDevServer::Port ([in] long)</td>
<td>This is Port, a member of class IDevServer.</td>
</tr>
<tr>
<td>IDevServer::Port ([out, retval] long*)</td>
<td>This is Port, a member of class IDevServer.</td>
</tr>
</tbody>
</table>

**Group**

**IDeveloper Methods**

**IDL**

```idl
[propget, id(0x000000CA)]
HRESULT _stdcall Port([out, retval] long* Value);
```

**Description**

This is Port, a member of class IDevServer.

**Group**

**Port Method**

**IDL**

```idl
[propput, id(0x000000CA)]
HRESULT _stdcall Port([in] long Value);
```

**Description**

This is Port, a member of class IDevServer.

**Group**

**Port Method**

9.1.1.7.1.3 StartBrowser Method

Instructs VirtualUI whether start or not the local web browser upon VirtualUI activation.

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDevServer::StartBrowser ([in] VARIANT_BOOL)</td>
<td>This is StartBrowser, a member of class IDevServer.</td>
</tr>
<tr>
<td>IDevServer::StartBrowser ([out, retval]</td>
<td>This is StartBrowser, a member of class IDevServer.</td>
</tr>
</tbody>
</table>
### Group

**IDevServer Methods**

**IDL**

```
[propget, id(0x000000CB)]
HRESULT _stdcall StartBrowser([out, retval] VARIANT_BOOL* Value);
```

**Description**

This is StartBrowser, a member of class IDevServer.

### Group

**StartBrowser Method**

**IDL**

```
[propput, id(0x000000CB)]
HRESULT _stdcall StartBrowser([in] VARIANT_BOOL Value);
```

**Description**

This is StartBrowser, a member of class IDevServer.

### Group

**StartBrowser Method**

9.1.1.8 **IClientSettings Interface**

Contains settings that are synchronized with the VirtualUI Viewer.

**Class Hierarchy**

```
IDispatch  VIRTUALUIIClientSettings
```

**IDL**

```
[ uuid(439624CA-ED33-47BE-9211-91290F29584A), dual, oleautomation ]
interface IClientSettings : IDispatch;
```

**File**

VirtualUIX.ridl

**Library**

VirtualUI Runtime Library
9.1.1.8.1 IClientSettings Methods

The methods of the IClientSettings class are listed here.

## Interface
IClientSettings Interface

## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CursorVisible</td>
<td>Hides/shows the mouse pointer.</td>
</tr>
<tr>
<td>MouseMoveGestureAction</td>
<td>Specifies whether the &quot;mouse move&quot; simulation on a touch device is interpreted as a mouse move or as a mouse wheel.</td>
</tr>
<tr>
<td>MouseMoveGestureStyle</td>
<td>Valid for touch devices. Specifies whether the mouse pointer is shown and acts on the exact spot of the finger touch (absolute) or its position is managed relatively to the movement of the finger touch (relative).</td>
</tr>
</tbody>
</table>

### 9.1.1.8.1.1 CursorVisible Method

Hides/shows the mouse pointer.

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IClientSettings::CursorVisible ([in] VARIANT_BOOL)</td>
<td>This is CursorVisible, a member of class IClientSettings.</td>
</tr>
<tr>
<td>IClientSettings::CursorVisible ([out, retval] VARIANT_BOOL*)</td>
<td>This is CursorVisible, a member of class IClientSettings.</td>
</tr>
</tbody>
</table>

## Group
IClientSettings Methods

### IDL

```idl
[propget, id(0x000000CB)]
HRESULT _stdcall CursorVisible([out, retval] VARIANT_BOOL* Value);
```

## Description

This is CursorVisible, a member of class IClientSettings.

## Group

CursorVisible Method
IDL
[propput, id(0x000000CB)]
HRESULT _stdcall CursorVisible([in] VARIANT_BOOL Value);

Description
This is CursorVisible, a member of class IClientSettings.

Group
CursorVisible Method

9.1.1.8.1.2 MouseMoveGestureAction Method

Specifies whether the "mouse move" simulation on a touch device is interpreted as a mouse move or as a mouse wheel.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IClientSettings::MouseMoveGestureAction ([in] enum MouseMoveGestureAction)</td>
<td>This is MouseMoveGestureAction, a member of class IClientSettings.</td>
</tr>
<tr>
<td>IClientSettings::MouseMoveGestureAction ([out, retval] enum MouseMoveGestureAction*)</td>
<td>This is MouseMoveGestureAction, a member of class IClientSettings.</td>
</tr>
</tbody>
</table>

Group
IClientSettings Methods

IDL
[propput, id(0x000000CA)]
HRESULT _stdcall MouseMoveGestureAction([in] enum MouseMoveGestureAction Value);

Description
This is MouseMoveGestureAction, a member of class IClientSettings.

Group
MouseMoveGestureAction Method

IDL
[propput, id(0x000000CA)]
HRESULT _stdcall MouseMoveGestureAction([in] enum MouseMoveGestureAction Value);
This is MouseMoveGestureAction, a member of class IClientSettings.

**Group**

MouseMoveGestureAction Method

9.1.1.8.1.3 MouseMoveGestureStyle Method

Valid for touch devices. Specifies whether the mouse pointer is shown and acts on the exact spot of the finger touch (absolute) or its position is managed relatively to the movement of the finger touch (relative).

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IClientSettings::MouseMoveGestureStyle ([in] enum MouseMoveGestureStyle)</code></td>
<td>This is MouseMoveGestureStyle, a member of class IClientSettings.</td>
</tr>
<tr>
<td><code>IClientSettings::MouseMoveGestureStyle ([out, retval] enum MouseMoveGestureStyle*)</code></td>
<td>This is MouseMoveGestureStyle, a member of class IClientSettings.</td>
</tr>
</tbody>
</table>

**Group**

IClientSettings Methods

**IDL**

```idl
[propput, id(0x000000C9)]
HRESULT _stdcall MouseMoveGestureStyle([in] enum MouseMoveGestureStyle Value);
```

**Description**

This is MouseMoveGestureStyle, a member of class IClientSettings.

**Group**

MouseMoveGestureStyle Method

**IDL**

```idl
[propput, id(0x000000C9)]
HRESULT _stdcall MouseMoveGestureStyle([in] enum MouseMoveGestureStyle Value);
```

**Description**

This is MouseMoveGestureStyle, a member of class IClientSettings.
9.1.1.9 IJSValue Interface

Base interface for IJSNamedValue, IJSPROPERTY and IJSArgument.

Class Hierarchy

IDL
[ uuid(6DE2E6A0-3C3A-47DC-9A93-928135EDAC90), dual, oleautomation ]
interface IJSValue : IDispatch;

File
VirtualUI.ridl

Library
VirtualUI Runtime Library

9.1.1.9.1 IJSValue Methods

The methods of the IJSValue class are listed here.

Interface
IJSValue Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsBool</td>
<td>Sets/gets a boolean value.</td>
</tr>
<tr>
<td>AsFloat</td>
<td>Sets/gets a float value.</td>
</tr>
<tr>
<td>AsInt</td>
<td>Sets/gets an integer value.</td>
</tr>
<tr>
<td>AsJSON</td>
<td>Sets/gets a JSON-formatted string value.</td>
</tr>
<tr>
<td>AsString</td>
<td>Sets/gets a string value.</td>
</tr>
<tr>
<td>DataType</td>
<td>Sets/gets a data type.</td>
</tr>
<tr>
<td>RawValue</td>
<td>This is the overview for the RawValue method overload.</td>
</tr>
</tbody>
</table>

9.1.1.9.1.1 AsBool Method

Sets/gets a boolean value.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
IJSValue::AsBool ([in] VARIANT_BOOL)
This is AsBool, a member of class IJSValue.

IJSValue::AsBool ([out, retval] VARIANT_BOOL*)
This is AsBool, a member of class IJSValue.

Group
IJSValue Methods

IDL
[propget, id(0x000000CD)]
HRESULT _stdcall AsBool([out, retval] VARIANT_BOOL* Value);

Description
This is AsBool, a member of class IJSValue.

Group
AsBool Method

IDL
[propput, id(0x000000CD)]
HRESULT _stdcall AsBool([in] VARIANT_BOOL Value);

Description
This is AsBool, a member of class IJSValue.

Group
AsBool Method

9.1.1.9.1.2 AsFloat Method
Sets/gets a float value.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSValue::AsFloat ([in] float)</td>
<td>This is AsFloat, a member of class IJSValue.</td>
</tr>
<tr>
<td>IJSValue::AsFloat ([out, retval] float*)</td>
<td>This is AsFloat, a member of class IJSValue.</td>
</tr>
</tbody>
</table>

Group
IJSValue Methods
### IDL

```idl
[propget, id(0x000000CE)]
HRESULT _stdcall AsFloat([out, retval] float* Value);
```

#### Description

This is AsFloat, a member of class IJSValue.

#### Group

AsFloat Method

### IDL

```idl
[propput, id(0x000000CE)]
HRESULT _stdcall AsFloat([in] float Value);
```

#### Description

This is AsFloat, a member of class IJSValue.

#### Group

AsFloat Method

---

#### 9.1.1.9.1.3 AsInt Method

Sets/gets an integer value.

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>† IJSValue::AsInt ([in] long)</td>
<td>This is AsInt, a member of class IJSValue.</td>
</tr>
<tr>
<td>† IJSValue::AsInt ([out, retval] long*)</td>
<td>This is AsInt, a member of class IJSValue.</td>
</tr>
</tbody>
</table>

#### Group

IJSValue Methods

### IDL

```idl
[propget, id(0x000000CC)]
HRESULT _stdcall AsInt([out, retval] long* Value);
```

#### Description

This is AsInt, a member of class IJSValue.

#### Group

IJSValue Methods
AsInt Method

IDL
[propput, id(0x000000CC)]
HRESULT _stdcall AsInt([in] long Value);

Description
This is AsInt, a member of class IJSValue.

Group
AsInt Method

9.1.9.1.4 AsJSON Method

Sets/gets a JSON-formatted string value.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSValue::AsJSON ([in] BSTR)</td>
<td>This is AsJSON, a member of class IJSValue.</td>
</tr>
<tr>
<td>IJSValue::AsJSON ([out, retval] BSTR*)</td>
<td>This is AsJSON, a member of class IJSValue.</td>
</tr>
</tbody>
</table>

Group
IJSValue Methods

IDL
[propput, id(0x000000CF)]
HRESULT _stdcall AsJSON([out, retval] BSTR* Value);

Description
This is AsJSON, a member of class IJSValue.

Group
AsJSON Method

IDL
[propput, id(0x000000CF)]
HRESULT _stdcall AsJSON([in] BSTR Value);

Description
This is AsJSON, a member of class IJSValue.
## Group

### AsJSON Method

#### 9.1.1.9.1.5 AsString Method

Sets/gets a string value.

<table>
<thead>
<tr>
<th>Overload List</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>IJSValue::AsString ([in] BSTR)</td>
</tr>
<tr>
<td>IJSValue::AsString ([out, retval] BSTR*)</td>
</tr>
</tbody>
</table>

### Group

#### IJSValue Methods

#### IDL

[propget, id(0x000000CB)]

HRESULT _stdcall AsString([out, retval] BSTR* Value);

#### Description

This is AsString, a member of class IJSValue.

### Group

#### AsString Method

#### IDL

[propput, id(0x000000CB)]

HRESULT _stdcall AsString([in] BSTR Value);

#### Description

This is AsString, a member of class IJSValue.

### Group

#### AsString Method

#### 9.1.1.9.1.6 DataType Method

Sets/gets a data type.
Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSValue::DataType ([in] enum IJSDataType)</td>
<td>This is DataType, a member of class IJSValue.</td>
</tr>
<tr>
<td>IJSValue::DataType ([out, retval] enum IJSDataType*)</td>
<td>This is DataType, a member of class IJSValue.</td>
</tr>
</tbody>
</table>

Group

IJSValue Methods

**IDL**

[propput, id(0x000000C9)]

HRESULT _stdcall DataType([in] enum IJSDataType Value);

Description

This is DataType, a member of class IJSValue.

Group

DataType Method

**IDL**

[propput, id(0x000000C9)]

HRESULT _stdcall DataType([in] enum IJSDataType Value);

Description

This is DataType, a member of class IJSValue.

Group

DataType Method

9.1.1.9.1.7 RawValue Method

This is the overview for the RawValue method overload.

Description

Sets/gets the rawvalue.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSValue::RawValue ([in] VARIANT)</td>
<td>This is RawValue, a member of class IJSValue.</td>
</tr>
</tbody>
</table>
This is RawValue, a member of class IJSValue.

**Group**

**IJSValue Methods**

**IDL**

```
[propget, id(0x000000CA)]
HRESULT _stdcall RawValue([out, retval] VARIANT* Value);
```

**Description**

This is RawValue, a member of class IJSValue.

**Group**

**RawValue Method**

**IDL**

```
[propput, id(0x000000CA)]
HRESULT _stdcall RawValue([in] VARIANT Value);
```

**Description**

This is RawValue, a member of class IJSValue.

**Group**

**RawValue Method**

9.1.1.10 **IJSNamedValue Interface**

Base interface for IJSProperty and IJSArgument.

**Class Hierarchy**

```
IDispatch  IJSValue  VirtualUIJSNamedValue
```

**IDL**

```
[ uuid(E492419B-00AC-4A91-9AE9-9A82B07E89AE), dual, oleautomation ]
interface IJSNamedValue : IJSValue;
```

**File**

VirtualUIX.ridl

**Library**

VirtualUI Runtime Library
9.1.1.10.1  IJSNamedValue Methods

The methods of the IJSNamedValue class are listed here.

 Interface

IJSNamedValue Interface

 Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Sets/gets the name.</td>
</tr>
</tbody>
</table>

9.1.1.10.1.1  Name Method

Sets/gets the name.

 Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name ([in] BSTR)</td>
<td>This is Name, a member of class IJSNamedValue.</td>
</tr>
<tr>
<td>Name ([out, retval] BSTR*)</td>
<td>This is Name, a member of class IJSNamedValue.</td>
</tr>
</tbody>
</table>

 Group

IJSNamedValue Methods

IDL
[propget, id(0x0000012D)]
HRESULT _stdcall Name([out, retval] BSTR* Value);

Description
This is Name, a member of class IJSNamedValue.

 Group

Name Method

IDL
[propput, id(0x0000012D)]
HRESULT _stdcall Name([in] BSTR Value);

Description
This is Name, a member of class IJSNamedValue.
### Group

**Name Method**

### 9.1.1.11 IJSObject Interface

Represents a custom remotable object.

#### Class Hierarchy

- [uuid(59342310-79A7-4B14-8B63-6DF05609AE30), dual, oleautomation]
  - **interface** IJSObject : IDispatch;

#### File

VirtualUIX.ridl

#### Remarks

IJSObject allows you to define an object model that is mirrored on the client side, and allows for an easy, powerful and straight-forward way to connect the web browser client application and the remoted Windows application.

IJSObject can contain properties (IJSProperties), methods (IJSMethods), events (IJSEvents) and children objects. Changes to properties values are propagated in from server to client and vice versa, keeping the data synchronized.

IJSObject is defined as a model seen from the client perspective. A method (IJSMethod) is called on the client side and executed on the server side. An event (IJSEvent) is called on the server side and raised on the client side.

#### Library

**VirtualUI Runtime Library**

### 9.1.1.11.1 IJSObject Methods

The methods of the IJSObject class are listed here.

#### Interface

**IJSObject Interface**

#### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Id" alt="Id" /></td>
<td>This is the overview for the Id method overload.</td>
</tr>
<tr>
<td><img src="Properties" alt="Properties" /></td>
<td>This is Properties, a member of class IJSObject.</td>
</tr>
</tbody>
</table>
9.1.11.1.1 Id Method

This is the overview for the Id method overload.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSObject::Id ([in] BSTR)</td>
<td>This is Id, a member of class IJSObject.</td>
</tr>
<tr>
<td>IJSObject::Id ([out, retval] BSTR*)</td>
<td>This is Id, a member of class IJSObject.</td>
</tr>
</tbody>
</table>

**Group**

IJSObject Methods

**IDL**

[propget, id(0x000000C9)]
HRESULT _stdcall Id([out, retval] BSTR* Value);

**Description**

This is Id, a member of class IJSObject.

**Group**

Id Method

**IDL**

[propput, id(0x000000C9)]
HRESULT _stdcall Id([in] BSTR Value);

**Description**

This is Id, a member of class IJSObject.
9.1.1.11.1.2 IJSObject::Properties Method

**IDL**

```idl
[propget, id(0x000000CA)]
HRESULT _stdcall Properties([out, retval] IJSProperties** Value);
```

**Description**

This is Properties, a member of class IJSObject.

**Group**

IJSObject Methods

9.1.1.11.1.3 IJSObject::Methods Method

**IDL**

```idl
[propget, id(0x000000CB)]
HRESULT _stdcall Methods([out, retval] IJSMethods** Value);
```

**Description**

This is Methods, a member of class IJSObject.

**Group**

IJSObject Methods

9.1.1.11.1.4 IJSObject::Events Method

**IDL**

```idl
[propget, id(0x000000CC)]
HRESULT _stdcall Events([out, retval] IJSEvents** Value);
```

**Description**

This is Events, a member of class IJSObject.

**Group**

IJSObject Methods

9.1.1.11.1.5 IJSObject::Objects Method

**IDL**

```idl
[propget, id(0x000000CD)]
HRESULT _stdcall Objects([out, retval] IJSObjects** Value);
```

**Description**
This is Objects, a member of class IJSObject.

**Group**

**IJSObject Methods**

### 9.1.11.1.6 IJSObject::FireEvent Method

**IDL**

```idl
[id(0x000000CE)]
HRESULT _stdcall FireEvent([in] BSTR Name, [in] IJSArguments* Arguments);
```

**Description**

This is FireEvent, a member of class IJSObject.

**Group**

**IJSObject Methods**

### 9.1.11.1.7 IJSObject::ApplyChanges Method

**IDL**

```idl
[id(0x000000CF)]
HRESULT _stdcall ApplyChanges();
```

**Description**

This is ApplyChanges, a member of class IJSObject.

**Group**

**IJSObject Methods**

### 9.1.11.1.8 IJSObject::ApplyModel Method

**IDL**

```idl
[id(0x000000D0)]
HRESULT _stdcall ApplyModel();
```

**Description**

This is ApplyModel, a member of class IJSObject.
9.1.1.12 IJSObjects Interface

Collection of IJSObjects

Class Hierarchy

IDL
[ uuid(C2406011-568E-4EAC-B95C-EF29E4806B86), dual, oleautomation ]
interface IJSObjects : IDispatch;

File
VirtualUIX.ridl

Library
VirtualUI Runtime Library

9.1.1.12.1 IJSObjects Methods

The methods of the IJSObjects class are listed here.

Interface
IJSObjects Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Returns the number of items in the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Returns an item.</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the collection.</td>
</tr>
<tr>
<td>Add</td>
<td>Adds a new object to the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an item from the collection.</td>
</tr>
</tbody>
</table>

9.1.1.12.1.1 IJSObjects::Count Method

Returns the number of items in the collection.

IDL
[propget, id(0x000000C9)]
HRESULT _stdcall Count([out, retval] long* Value);

Group
IJSObjects Methods

9.1.1.12.1.2 IJSObjects::Item Method

Returns an item.

**IDL**

```idl
[propget, id(0x00000000), defaultcollelem]
HRESULT _stdcall Item([in] VARIANT Index, [out, retval] IJSObject** Value);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] VARIANT Index</td>
<td>OleVariant. Number indicating the index of the item in the collection or string indicating the name of the item.</td>
</tr>
<tr>
<td>OutRetVal</td>
<td>Returns the selected item.</td>
</tr>
</tbody>
</table>

**Group**

IJSObjects Methods

9.1.1.12.1.3 IJSObjects::Clear Method

Clears the collection.

**IDL**

```idl
[id(0x000000CC)]
HRESULT _stdcall Clear();
```

**Group**

IJSObjects Methods

9.1.1.12.1.4 IJSObjects::Add Method

Adds a new object to the collection.

**IDL**

```idl
[id(0x000000CD)]
HRESULT _stdcall Add([in] BSTR Id, [out, retval] IJSObject** Value);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Object name.</td>
</tr>
<tr>
<td>OutRetVal</td>
<td>Returns the newly created object.</td>
</tr>
</tbody>
</table>

**Group**
9.1.1.12.1.5 IJSObjects::Remove Method

Removes an item from the collection.

**IDL**

```
[id(0x000000CE)]
HRESULT _stdcall Remove([in] IJSObject* Item);
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSObject* Item</td>
<td>Item to remove</td>
</tr>
</tbody>
</table>

**Group**

IJSObjects Methods

9.1.1.13 IJSProperty Interface

Represents a property in a custom remotelable object.

**Class Hierarchy**

```
IDispatch -> VirtualUIJSValue -> VirtualUIJSNamedValue -> VirtualUIJSPROPERTY
```

**IDL**

```
[uuid(1F95C0E9-E7BF-48C9-AA35-88AD0149109B), dual, oleautomation]
interface IJSProperty : IJSNamedValue;
```

**File**

VirtualUIX.ridl

**Library**

VirtualUI Runtime Library

9.1.1.13.1 IJSProperty Methods

The methods of the IJSProperty class are listed here.

**Interface**

IJSProperty Interface

**Public Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
9.1.1.13.1.1  ReadOnly Method

Sets/gets the readonly attribute. If Readonly is true, the property value cannot be altered on the client side.

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSProperty::ReadOnly ([in] VARIANT_BOOL)</td>
<td>This is ReadOnly, a member of class IJSProperty.</td>
</tr>
<tr>
<td>IJSProperty::ReadOnly ([out, retval] VARIANT_BOOL*)</td>
<td>This is ReadOnly, a member of class IJSProperty.</td>
</tr>
</tbody>
</table>

### Group

#### IJSProperty Methods

**IDL**

[propget, id(0x00000191)]

HRESULT _stdcall ReadOnly([out, retval] VARIANT_BOOL* Value);

**Description**

This is ReadOnly, a member of class IJSProperty.

#### Group

**ReadOnly Method**

**IDL**

[propput, id(0x00000191)]

HRESULT _stdcall ReadOnly([in] VARIANT_BOOL Value);

**Description**

This is ReadOnly, a member of class IJSProperty.
9.1.1.13.1.2 IJSProperty::OnGet Method

Allows you to pass an interface that will receive control at property initialization and when you make a call to IJSObject:ApplyChanges.

**IDL**

```idl
[id(0x00000192)]
HRESULT __stdcall OnGet([in] IJSBinding* Binding, [out, retval] IJSProperty** OutRetVal);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSBinding* Binding</td>
<td>IJSBinding interface</td>
</tr>
<tr>
<td>[out, retval] IJSProperty** OutRetVal</td>
<td>Returns the parent property</td>
</tr>
</tbody>
</table>

**Group**

IJSProperty Methods

9.1.1.13.1.3 IJSProperty::OnSet Method

Allows you to pass an interface that will receive control when a property changes on the client side.

**IDL**

```idl
[id(0x00000193)]
HRESULT __stdcall OnSet([in] IJSBinding* Binding, [out, retval] IJSProperty** OutRetVal);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSBinding* Binding</td>
<td>IJSBinding interface</td>
</tr>
<tr>
<td>[out, retval] IJSProperty** OutRetVal</td>
<td>Returns the parent property</td>
</tr>
</tbody>
</table>

**Group**

IJSProperty Methods

9.1.1.14 IJSProperties Interface

Collection of properties.

**Class Hierarchy**

[Diagram showing the class hierarchy]

**IDL**

```idl
[ uuid(FCBB688F-8FB2-42C1-86FC-0AAF3B2A500C), dual, oleautomation ]
interface IJSProperties : IDispatch;
```

**File**

© 2015, Cybele Software, Inc.
9.1.1.14.1 IJSProperties Methods

The methods of the IJSProperties class are listed here.

### Interface

**IJSProperties Interface**

#### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Returns the number of items in the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Returns an item.</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the collection.</td>
</tr>
<tr>
<td>Add</td>
<td>Adds a new property to the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an item from the collection.</td>
</tr>
</tbody>
</table>

#### 9.1.1.14.1.1 IJSProperties::Count Method

Returns the number of items in the collection.

**IDL**

```
[propget, id(0x000000C9)]
HRESULT _stdcall Count([out, retval] long* Value);
```

#### 9.1.1.14.1.2 IJSProperties::Item Method

Returns an item.

**IDL**

```
[propget, id(0x00000000), defaultcollelem]
HRESULT _stdcall Item([in] VARIANT Index, [out, retval] IJSProperty** Value);
```

#### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
## IJSProperties Methods

### 9.1.1.14.1.3 IJSProperties::Clear Method

Clears the collection.

**IDL**

```idl
[id(0x000000CC)]
HRESULT _stdcall Clear();
```

### 9.1.1.14.1.4 IJSProperties::Add Method

Adds a new property to the collection.

**IDL**

```idl
[id(0x000000CD)]
HRESULT _stdcall Add([in] BSTR Name, [out, retval] IJSProperty** OutRetVal);
```

#### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR Name</td>
<td>Property name.</td>
</tr>
<tr>
<td>[out, retval] IJSProperty**</td>
<td>Returns the newly created property.</td>
</tr>
</tbody>
</table>

### 9.1.1.14.1.5 IJSProperties::Remove Method

Removes an item from the collection.

**IDL**

```idl
[id(0x000000CE)]
HRESULT _stdcall Remove([in] IJSProperty* Item);
```

#### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSProperty* Item</td>
<td></td>
</tr>
</tbody>
</table>
9.1.1.15 IJSArgument Interface

Represents a parameter in events and methods.

**Class Hierarchy**

```
IDispatch → VirtualUI:IJSValue → VirtualUI:IJSNamedValue → VirtualUI:IJSArgument
```

IDL

```
[ uuid(8F8C4462-D7B5-4696-BAD5-16DFAA6E2601), dual, oleautomation ]
interface IJSArgument : IJSNamedValue;
```

**File**

VirtualUIX.ridl

**Library**

VirtualUI Runtime Library

9.1.1.16 IJSArguments Interface

Collection of IJSArgument.

**Class Hierarchy**

```
IDispatch → VirtualUI:IJSArguments
```

IDL

```
[ uuid(FC097EF5-6D8A-4C80-A2AD-382FDC75E901), dual, oleautomation ]
interface IJSArguments : IDispatch;
```

**File**

VirtualUIX.ridl

**Library**

VirtualUI Runtime Library

9.1.1.16.1 IJSArguments Methods

The methods of the IJSArguments class are listed here.
IJSArguments Interface

### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Returns the number of items in the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Returns an item of the collection.</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the collection.</td>
</tr>
<tr>
<td>Add</td>
<td>Adds an argument to the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an item from the collection.</td>
</tr>
</tbody>
</table>

#### 9.1.16.1.1 IJSArguments::Count Method

Returns the number of items in the collection.

**IDL**

```idl
[propget, id(0x000000C9)]
HRESULT _stdcall Count([out, retval] long* Value);
```

#### Group

**IJSArguments Methods**

#### 9.1.16.1.2 IJSArguments::Item Method

Returns an item of the collection.

**IDL**

```idl
[propget, id(0x00000000), defaultcollelem]
HRESULT _stdcall Item([in] VARIANT Index, [out, retval] IJSArgument** Value);
```

#### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] VARIANT Index</td>
<td>OleVariant. Number indicating the index in the collection or string indicating the name of the method.</td>
</tr>
<tr>
<td>OutRetVal</td>
<td>Returns the selected IJSMethod</td>
</tr>
</tbody>
</table>

#### Group

**IJSArguments Methods**

#### 9.1.16.1.3 IJSArguments::Clear Method

Clears the collection.
IDL
[id(0x000000CC)]
HRESULT _stdcall Clear();

_GROUP_

**IJSArguments Methods**

9.1.16.1.4 **IJSArguments::Add Method**

Adds an argument to the collection.

IDL
[id(0x000000CD)]
HRESULT _stdcall Add([in] BSTR Name, [out, retval] IJSArgument** OutRetVal);

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR Name</td>
<td>Argument name</td>
</tr>
<tr>
<td>[out, retval] IJSArgument** OutRetVal</td>
<td>Returns a newly created <strong>IJSArgument</strong></td>
</tr>
</tbody>
</table>

**Group**

**IJSArguments Methods**

9.1.16.1.5 **IJSArguments::Remove Method**

Removes an item from the collection.

IDL
[id(0x000000CE)]
HRESULT _stdcall Remove([in] IJSArgument* Item);

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSArgument* Item</td>
<td><strong>IJSMethod</strong> to remove</td>
</tr>
</tbody>
</table>

**Group**

**IJSArguments Methods**

9.1.17 **IJSMethod Interface**

This interface represents a method called on the client side and executed on the server side.

**Class Hierarchy**

[**Dispach**]→[VirtualUI::**IJSMethod**]
IDL
[ uuid(C45D6A8F-AD4A-47BB-AC3A-C125D6D5D27E), dual, oleautomation ]
interface IJSMethod : IDispatch;

File
VirtualUIX.ridl

Library
VirtualUI Runtime Library

9.1.1.17.1 IJSMethod Methods

The methods of the IJSMethod class are listed here.

Interface
IJSMethod Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>This is the overview for the Name method overload.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Collection of arguments</td>
</tr>
<tr>
<td>ReturnValue</td>
<td>Data to return to the remote caller.</td>
</tr>
<tr>
<td>AddArgument</td>
<td>Adds an argument to the parameters list</td>
</tr>
<tr>
<td>OnCall</td>
<td>Allows you to pass a IJSCallback interface that will receive a the remote call.</td>
</tr>
</tbody>
</table>

9.1.1.17.1.1 Name Method

This is the overview for the Name method overload.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSMethod::Name ([in] BSTR)</td>
<td>This is Name, a member of class IJSMethod.</td>
</tr>
<tr>
<td>IJSMethod::Name ([out, retval] BSTR*)</td>
<td>This is Name, a member of class IJSMethod.</td>
</tr>
</tbody>
</table>

Group
IJSMethod Methods

IDL
[propget, id(0x000000C9)]
HRESULT _stdcall Name([out, retval] BSTR* Value);

**Description**
This is Name, a member of class IJSMethod.

**Group**
Name Method

**IDL**
[propput, id(0x000000C9)]
HRESULT _stdcall Name([in] BSTR Value);

**Description**
This is Name, a member of class IJSMethod.

**Group**
Name Method

9.1.1.17.1.2  IJSMethod::Arguments Method
Collection of arguments

**IDL**
[proppget, id(0x000000CA)]
HRESULT _stdcall Arguments([out, retval] IJSArguments** Value);

**Group**
IJSMethod Methods

9.1.1.17.1.3  IJSMethod::ReturnValue Method
Data to return to the remote caller.

**IDL**
[proppget, id(0x000000CB)]
HRESULT _stdcall ReturnValue([out, retval] IJSValue** Value);

**Group**
IJSMethod Methods

9.1.1.17.1.4  IJSMethod::AddArgument Method
Adds an argument to the parameters list
IDL
[id(0x000000CC)]
HRESULT _stdcall AddArgument([in] BSTR Name, [in] enum IJSDataType DataType, [out, retval] IJSMethod** OutRetVal);

 Parameters
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR Name</td>
<td>Argument name</td>
</tr>
<tr>
<td>[in] enum IJSDataType DataType</td>
<td>Data type.</td>
</tr>
<tr>
<td>[out, retval] IJSMethod** OutRetVal</td>
<td>Returns the newly created argument.</td>
</tr>
</tbody>
</table>

 Group
IJSMethod Methods

9.1.1.17.1.5 IJSMethod::OnCall Method

Allows you to pass a IJSCallback interface that will receive a the remote call.

IDL
[id(0x000000CD)]
HRESULT _stdcall OnCall([in] IJSCallback* Callback, [out, retval] IJSMethod** OutRetVal);

 Parameters
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSCallback* Callback</td>
<td>Callback interface.</td>
</tr>
<tr>
<td>[out, retval] IJSMethod** OutRetVal</td>
<td>Returns the parent IJSMethod.</td>
</tr>
</tbody>
</table>

 Group
IJSMethod Methods

9.1.1.18 IJSMethods Interface

Collection of methods.

 Class Hierarchy

IDL
[ uuid(E4CB461F-586E-4121-ABD7-345B87BC423A), dual, oleautomation ]
interface IJSMethods : IDispatch;

 File
VirtualUIX.ridl

 Library
© 2015, Cybele Software, Inc.
VirtualUI Runtime Library

9.1.1.18.1 IJSMMethods Methods

The methods of the IJSMMethods class are listed here.

[Interface]

IJSMethods Interface

[Public Methods]

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Returns the number of items in the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Returns an item.</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the collection.</td>
</tr>
<tr>
<td>Add</td>
<td>Adds a new method to the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an item from the collection.</td>
</tr>
</tbody>
</table>

9.1.1.18.1.1 IJSMethods::Count Method

Returns the number of items in the collection.

**IDL**

[propget, id(0x000000C9)]

HRESULT _stdcall Count([out, retval] long* Value);

[Group]

IJSMethods Methods

9.1.1.18.1.2 IJSMethods::Item Method

Returns an item.

**IDL**

[propget, id(0x00000000), defaultcollelem]

HRESULT _stdcall Item([in] VARIANT Index, [out, retval] IJSMethod** Value);

[Parameters]

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] VARIANT Index</td>
<td>OleVariant. Number indicating the index of the item in the collection or string indicating the name of the item.</td>
</tr>
<tr>
<td>OutRetVal</td>
<td>Returns the selected item.</td>
</tr>
</tbody>
</table>
9.1.1.18.1.3 IJSMethods::Clear Method

Clears the collection.

**IDL**

```
[id(0x000000CC)]
HRESULT _stdcall Clear();
```

9.1.1.18.1.4 IJSMethods::Add Method

Adds a new method to the collection.

**IDL**

```
[id(0x000000CD)]
HRESULT _stdcall Add([in] BSTR Name, [out, retval] IJSMethod** OutRetVal);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR Name</td>
<td>Method name.</td>
</tr>
<tr>
<td>[out, retval] IJSMethod** OutRetVal</td>
<td>Returns the newly created method.</td>
</tr>
</tbody>
</table>

9.1.1.18.1.5 IJSMethods::Remove Method

Removes an item from the collection.

**IDL**

```
[id(0x000000CE)]
HRESULT _stdcall Remove([in] IJSMethod* Item);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSMethod* Item</td>
<td>Item to remove</td>
</tr>
</tbody>
</table>
9.1.1.19 IJSEvent Interface

This interface represents an event fired on the server side and raised on the client side.

Class Hierarchy

IDL

[ uuid(8B66EACD-9619-43CF-9196-DCDA17F5500E), dual, oleautomation ]

interface IJSEvent : IDispatch;

File

VirtualUIX.ridl

Library

VirtualUI Runtime Library

9.1.1.19.1 IJSEvent Methods

The methods of the IJSEvent class are listed here.

Interface

IJSEvent Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>This is the overview for the Name method overload.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Collection of arguments.</td>
</tr>
<tr>
<td>AddArgument</td>
<td>Adds an argument to the Arguments collection.</td>
</tr>
<tr>
<td>ArgumentAsNull</td>
<td>Sets the value of an argument to null.</td>
</tr>
<tr>
<td>ArgumentAsString</td>
<td>Sets the value of an argument with a string data type.</td>
</tr>
<tr>
<td>ArgumentAsInt</td>
<td>Sets the value of an argument with an integer data type.</td>
</tr>
<tr>
<td>ArgumentAsBool</td>
<td>Sets the value of an argument with a boolean data type.</td>
</tr>
<tr>
<td>ArgumentAsFloat</td>
<td>Sets the value of an argument with a float data type.</td>
</tr>
<tr>
<td>ArgumentAsJSON</td>
<td>Sets the value of an argument with a JSON formatted string.</td>
</tr>
<tr>
<td>Fire</td>
<td>Fires the event.</td>
</tr>
</tbody>
</table>
9.1.1.19.1.1 Name Method

This is the overview for the Name method overload.

- **Description**

  Sets/gets the event name.

- **Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSEvent::Name ([in] BSTR)</td>
<td>This is Name, a member of class IJSEvent.</td>
</tr>
<tr>
<td>IJSEvent::Name ([out, retval] BSTR*)</td>
<td>This is Name, a member of class IJSEvent.</td>
</tr>
</tbody>
</table>

- **Group**

  IJSEvent Methods

- **IDL**

  [propget, id(0x000000C9)]

  HRESULT _stdcall Name([out, retval] BSTR* Value);

- **Description**

  This is Name, a member of class IJSEvent.

- **Group**

  Name Method

- **IDL**

  [propput, id(0x000000C9)]

  HRESULT _stdcall Name([in] BSTR Value);

- **Description**

  This is Name, a member of class IJSEvent.

- **Group**

  Name Method

9.1.1.19.1.2 IJSEvent::Arguments Method

Collection of arguments.

- **IDL**

  [propget, id(0x000000CA)]
HRESULT _stdcall Arguments([out, retval] IJSArguments** Value);

**Group**

IJSEvent Methods

9.1.1.19.1.3 IJSEvent::AddArgument Method

Adds an argument to the **Arguments** collection.

**IDL**

[id(0x000000CB)]

HRESULT _stdcall AddArgument([in] BSTR Name, [in] enum IJSDataType DataType, [out, retval] IJSEvent** OutRetVal);

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR Name</td>
<td>Name of argument.</td>
</tr>
<tr>
<td>[in] enum IJSDataType DataType</td>
<td>Data type of the argument.</td>
</tr>
<tr>
<td>[out, retval] IJSEvent** OutRetVal</td>
<td>Returns the parent IJSEvent.</td>
</tr>
</tbody>
</table>

**Group**

IJSEvent Methods

9.1.1.19.1.4 IJSEvent::ArgumentAsNull Method

Sets the value of an argument to null.

**IDL**

[id(0x000000CC)]

HRESULT _stdcall ArgumentAsNull([in] VARIANT Index, [out, retval] IJSEvent** OutRetVal);

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] VARIANT Index</td>
<td>OleVariant. Number or string value identifying the argument by index or name.</td>
</tr>
<tr>
<td>[out, retval] IJSEvent** OutRetVal</td>
<td>Returns the parent IJSEvent.</td>
</tr>
</tbody>
</table>

**Group**

IJSEvent Methods

9.1.1.19.1.5 IJSEvent::ArgumentAsString Method

Sets the value of an argument with a string data type.

**IDL**

[id(0x000000CD)]
HRESULT _stdcall ArgumentAsString([in] VARIANT Index, [in] BSTR Value, [out, retval] IJSEvent** OutRetVal);

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] VARIANT Index</td>
<td>OleVariant. Number or string value identifying the argument by index or name.</td>
</tr>
<tr>
<td>[in] BSTR Value</td>
<td>String value.</td>
</tr>
<tr>
<td>[out, retval] IJSEvent** OutRetVal</td>
<td>Returns the parent IJSEvent.</td>
</tr>
</tbody>
</table>

** Group

IJSEvent Methods

9.1.19.1.6 IJSEvent::ArgumentAsInt Method

Sets the value of an argument with an integer data type.

IDL

[id(0x000000CE)]

HRESULT _stdcall ArgumentAsInt([in] VARIANT Index, [in] long Value, [out, retval] IJSEvent** OutRetVal);

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] VARIANT Index</td>
<td>OleVariant. Number or string value identifying the argument by index or name.</td>
</tr>
<tr>
<td>[in] long Value</td>
<td>Integer value.</td>
</tr>
<tr>
<td>[out, retval] IJSEvent** OutRetVal</td>
<td>Returns the parent IJSEvent.</td>
</tr>
</tbody>
</table>

** Group

IJSEvent Methods

9.1.19.1.7 IJSEvent::ArgumentAsBool Method

Sets the value of an argument with a boolean data type.

IDL

[id(0x000000CF)]

HRESULT _stdcall ArgumentAsBool([in] VARIANT Index, [in] VARIANT_BOOL Value, [out, retval] IJSEvent** OutRetVal);

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] VARIANT Index</td>
<td>OleVariant. Number or string value identifying the argument by index or name.</td>
</tr>
<tr>
<td>[in] VARIANT_BOOL Value</td>
<td>Boolean value.</td>
</tr>
<tr>
<td>[out, retval] IJSEvent** OutRetVal</td>
<td>Returns the parent IJSEvent.</td>
</tr>
</tbody>
</table>
## IJSEvent Methods

### 9.1.1.19.1.8 IJSEvent::ArgumentAsFloat Method

Sets the value of an argument with a float data type.

**IDL**

```idl
HRESULT _stdcall ArgumentAsFloat([in] VARIANT Index, [in] float Value, [out, retval] IJSEvent** OutRetVal);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] VARIANT Index</td>
<td>OleVariant. Number or string value identifying the argument by index or name.</td>
</tr>
<tr>
<td>[in] float Value</td>
<td>Float value.</td>
</tr>
<tr>
<td>[out, retval] IJSEvent** OutRetVal</td>
<td>Returns the parent IJSEvent.</td>
</tr>
</tbody>
</table>

### 9.1.1.19.1.9 IJSEvent::ArgumentAsJSON Method

Sets the value of an argument with a JSON formatted string.

**IDL**

```idl
HRESULT _stdcall ArgumentAsJSON([in] VARIANT Index, [in] BSTR Value, [out, retval] IJSEvent** OutRetVal);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] VARIANT Index</td>
<td>OleVariant. Number or string value identifying the argument by index or name.</td>
</tr>
<tr>
<td>[in] BSTR Value</td>
<td>String value with a valid JSON format.</td>
</tr>
<tr>
<td>[out, retval] IJSEvent** OutRetVal</td>
<td>Returns the parent IJSEvent.</td>
</tr>
</tbody>
</table>

### 9.1.1.19.1.10 IJSEvent::Fire Method

Fires the event.
9.1.1.20 IJSEvents Interface

Collection of events.

Class Hierarchy

```
IDL
[id(0x000000D2)]
HRESULT _stdcall Fire();
```

File

VirtualUIX.ridl

Library

VirtualUI Runtime Library

9.1.1.20.1 IJSEvents Methods

The methods of the IJSEvents class are listed here.

Interface

IJSEvents Interface

<table>
<thead>
<tr>
<th>Public Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>Returns the number of items in the collection.</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Returns an item.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Clears the collection.</td>
</tr>
<tr>
<td><strong>Add</strong></td>
<td>Adds an IJSEvent to the collection.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Removes an item from the collection.</td>
</tr>
</tbody>
</table>
9.1.1.20.1.1 IJSEvents::Count Method

Returns the number of items in the collection.

IDL
[propget, id(0x000000C9)]
HRESULT _stdcall Count([out, retval] long* Value);

Group
IJSEvents Methods

9.1.1.20.1.2 IJSEvents::Item Method

Returns an item.

IDL
[propget, id(0x00000000), defaultcollelem]
HRESULT _stdcall Item([in] VARIANT Index, [out, retval] IJSEvent** Value);

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] VARIANT Index</td>
<td>OleVariant. Number indicating the index of the item in the collection or string indicating the name of the item.</td>
</tr>
<tr>
<td>OutRetVal</td>
<td>Returns the selected item.</td>
</tr>
</tbody>
</table>

Group
IJSEvents Methods

9.1.1.20.1.3 IJSEvents::Clear Method

Clears the collection.

IDL
[id(0x000000CC)]
HRESULT _stdcall Clear();

Group
IJSEvents Methods

9.1.1.20.1.4 IJSEvents::Add Method

Adds an IJSEvent to the collection.

IDL
[id(0x000000CD)]
HRESULT _stdcall Add([in] BSTR Name, [out, retval] IJSEvent** OutRetVal);
### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR Name</td>
<td>Event name</td>
</tr>
<tr>
<td>[out, retval] IJSEvent** OutRetVal</td>
<td>Returns the newly created IJSEvent</td>
</tr>
</tbody>
</table>

### Group

**IJSEvents Methods**

#### 9.1.1.20.1.5 IJSEvents::Remove Method

Removes an item from the collection.

**IDL**

```idl
[id(0x000000CE)]
HRESULT _stdcall Remove([in] IJSEvent* Item);
```

### Group

**IJSEvents Methods**

#### 9.1.1.21 IJSBinding Interface

Used as a callback interface for binding external data source with the IJSProperty data.

**Class Hierarchy**

```
IDispatch -> VirtualUI.IJSBinding
```

**IDL**

```idl
[ uuid(ACFC2953-37F1-479E-B405-D0BB75E156E6), dual, oleautomation ]
interface IJSBinding : IDispatch;
```

**File**

VirtualUIX.ridl

**Library**

VirtualUI Runtime Library

#### 9.1.1.21.1 IJSBinding Methods

The methods of the IJSBinding class are listed here.
Interface

IJSBinding Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>Callback method.</td>
</tr>
</tbody>
</table>

9.1.1.21.1.1 IJSBinding::Set Method

Callback method.

IDL

```idl
[id(0x000000C9)]
HRESULT _stdcall Set([in] IJSObject* Parent, [in] IJSProperty* Prop);
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSObject* Parent</td>
<td>Parent IJSObject</td>
</tr>
<tr>
<td>[in] IJSProperty* Prop</td>
<td>Calling IJSProperty</td>
</tr>
</tbody>
</table>

Group

IJSBinding Methods

9.1.1.22 IJSCallback Interface

Used as a callback interface for executing the remote calling to IJSMethod.

Class Hierarchy

```idl
[uuid(ADD570A0-491A-4E40-8120-57B4D1245FD3), dual, oleautomation ]
interface IJSCallback : IDispatch;
```

File

VirtualUIX.ridl

Library

VirtualUI Runtime Library
9.1.1.22.1 IJSCallback Methods

The methods of the IJSCallback class are listed here.

- **Interface**
  IJSCallback Interface

- **Public Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback</td>
<td>Callback method.</td>
</tr>
</tbody>
</table>

9.1.1.22.1.1 IJSCallback::Callback Method

Callback method.

**IDL**

```idl
[id(0x000000C9)]
HRESULT _stdcall Callback([in] IJSObject* Parent, [in] IJSMethod* Method);
```

- **Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSObject* Parent</td>
<td>Parent IJSObject</td>
</tr>
<tr>
<td>Name</td>
<td>Calling method name</td>
</tr>
<tr>
<td>Arguments</td>
<td>IJSArguments</td>
</tr>
<tr>
<td>ReturnValue</td>
<td>Returned value to the client call.</td>
</tr>
</tbody>
</table>

- **Group**
  IJSCallback Methods

9.1.1.23 IEvents Interface

General VirtualUI events.

- **Class Hierarchy**
  VirtualUI::IEvents

**IDL**

```idl
[uuid(1C5700BC-2317-4062-B614-0A4E286CFE68)]
dispinterface IEvents;
```

- **File**
  VirtualUIX.ridl

© 2015, Cybele Software, Inc.
Library

VirtualUI Runtime Library

9.1.1.23.1 IEvents Methods

The methods of the IEvents class are listed here.

Interface

IEvents Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnGetUploadDir</td>
<td>Fires during an upload request, allowing you to change the save folder.</td>
</tr>
<tr>
<td>OnBrowserResize</td>
<td>Fires when the VirtualUI Viewer's container window resizes. Normally, when the browser resizes.</td>
</tr>
<tr>
<td>OnClose</td>
<td>Fires when the browser window is about to close.</td>
</tr>
<tr>
<td>OnReceiveMessage</td>
<td>Fires when a custom data string is sent from the web browser page.</td>
</tr>
<tr>
<td>OnDownloadEnd</td>
<td>Fires when the file has been sent.</td>
</tr>
</tbody>
</table>

9.1.1.23.1.1 IEvents::OnGetUploadDir Method

Fires during an upload request, allowing you to change the save folder.

IDL

```
[id(0x00000065)]
void OnGetUploadDir([in, out] BSTR* Directory, [in, out] VARIANT_BOOL* Handled);
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out] VARIANT_BOOL* Handled</td>
<td>[in,out] Indicates that the Directory was changed.</td>
</tr>
</tbody>
</table>

Group

IEvents Methods

9.1.1.23.1.2 IEvents::OnBrowserResize Method

Fires when the VirtualUI Viewer's container window resizes. Normally, when the browser resizes.

IDL
void OnBrowserResize([in, out] long* Width, [in, out] long* Height, [in, out] VARIANT_BOOL* ResizeMaximized);

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in, out] long* Width</td>
<td>Browser window width.</td>
</tr>
<tr>
<td>[in, out] long* Height</td>
<td>Browser window height.</td>
</tr>
<tr>
<td>Handled</td>
<td>[out] Returns whether to prevent the default processing.</td>
</tr>
</tbody>
</table>

**Remarks**

Allows you to take action when the VirtualUI Viewer's container window resizes. Set Handled to true to disable the default processing, which resizes all maximized windows.

**Group**

IEvents Methods

### 9.1.1.23.1.3 IEvents::OnClose Method

Fires when the browser window is about to close.

**IDL**

```idl
[id(0x00000067)]
void OnClose();
```

**Group**

IEvents Methods

### 9.1.1.23.1.4 IEvents::OnReceiveMessage Method

Fires when a custom data string is sent from the web browser page.

**IDL**

```idl
[id(0x00000068)]
void OnReceiveMessage([in] BSTR Data);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR Data</td>
<td>Data string</td>
</tr>
</tbody>
</table>

**Group**

IEvents Methods
9.1.1.23.1.5 IEvents::OnDownloadEnd Method

Fires when the file has been sent.

IDL
[ id(0x00000069) ]
void OnDownloadEnd([in] BSTR Filename);

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR Filename</td>
<td>File name.</td>
</tr>
</tbody>
</table>

Group

IEvents Methods

9.1.1.24 IJSObjectEvents Interface

Class Hierarchy

IDL
[ uuid(A3D640E8-CD18-4196-A1A2-C87C82B0F88B) ]
dispinterface IJSObjectEvents;

File

VirtualUIX.ridl

Description

This is class VirtualUI::IJSObjectEvents.

Library

VirtualUI Runtime Library

9.1.1.24.1 IJSObjectEvents Methods

The methods of the IJSObjectEvents class are listed here.

Interface

IJSObjectEvents Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

9.1.1.24.1.1 IJSObjectEvents::OnExecuteMethod Method

Fires when a method is called in the client side.

**IDL**

```
[id(0x00000065)]
void OnExecuteMethod([in] IJSObject* Caller, [in] IJSMethod* Method);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSObject* Caller</td>
<td>Object caller</td>
</tr>
<tr>
<td>Name</td>
<td>Method name</td>
</tr>
<tr>
<td>Arguments</td>
<td>Arguments</td>
</tr>
<tr>
<td>ReturnValue</td>
<td>Data to return to the client</td>
</tr>
</tbody>
</table>

**Group**

IJSObjectEvents Methods

9.1.1.24.1.2 IJSObjectEvents::OnPropertyChange Method

Fires when a property has changed on the client side.

**IDL**

```
[id(0x00000066)]
void OnPropertyChange([in] IJSObject* Caller, [in] IJSProperty* Prop);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IJSObject* Caller</td>
<td>Object caller</td>
</tr>
<tr>
<td>Name</td>
<td>Property name</td>
</tr>
<tr>
<td>Value</td>
<td>Property value</td>
</tr>
</tbody>
</table>

**Group**

IJSObjectEvents Methods

9.1.2 Server Configuration Library
### CoClasses

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>This is class VirtualUIS::Server.</td>
</tr>
</tbody>
</table>

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>This is record VirtualUIS::Protocol.</td>
</tr>
<tr>
<td>ProfileKind</td>
<td>This is record VirtualUIS::ProfileKind.</td>
</tr>
<tr>
<td>ScreenResolution</td>
<td>This is record VirtualUIS::ScreenResolution.</td>
</tr>
<tr>
<td>ServerSection</td>
<td>This is record VirtualUIS::ServerSection.</td>
</tr>
</tbody>
</table>

### Group

**ActiveX Interfaces**

### Interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IServer</td>
<td>Gives you access to server's settings</td>
</tr>
<tr>
<td>ILicense</td>
<td>Contains methods and properties to control VirtualUI Server's licence activation.</td>
</tr>
<tr>
<td>IProfile</td>
<td>A profile contains information about an application or web link configured to be opened in VirtualUI's home page (or directly through its URL).</td>
</tr>
<tr>
<td>IPDFiles</td>
<td>Contains the list of profiles registered in VirtualUI Server.</td>
</tr>
<tr>
<td>IBinding</td>
<td>Interface for the server's binding parameters.</td>
</tr>
<tr>
<td>ICertificate</td>
<td>Manages the certificate's configuration for HTTPS Binding.</td>
</tr>
<tr>
<td>IRDS</td>
<td>Manages the configuration of a Remote Desktop Services account.</td>
</tr>
<tr>
<td>IRDSEAccounnts</td>
<td>Contains a list of Remote Desktop Services accounts.</td>
</tr>
</tbody>
</table>

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWideString1</td>
<td>This is type VirtualUIS::PWideString1.</td>
</tr>
</tbody>
</table>

#### 9.1.2.1 VirtualUIS::Protocol Enumeration

IDL

```idl
[ uuid(D8C7713F-D823-4FC8-B361-B67667CBB9A9) ]
enum Protocol {
  PROTO_HTTP = 0,
```
### VirtualUIS::ProfileKind Enumeration

**IDL**

```
[ uuid(5B7C0230-6A31-468B-874A-65777244F4AF) ]
enum ProfileKind {
  PROFILE_APP = 0,
  PROFILE_WEBLINK = 1
};
```

**File**

VirtualUIS.ridl

**Description**

This is record VirtualUIS::ProfileKind.

**Library**

Server Configuration Library

### VirtualUIS::ScreenResolution Enumeration

**IDL**

```
[ uuid(F4DF441B-0C9A-40A3-A6F7-1B251E5765B3) ]
enum ScreenResolution {
  SCREENRES_Custom = 0,
  SCREENRES_FitToBrowser = 1,
  SCREENRES_FitToScreen = 2,
  SCREENRES_640x480 = 3,
  SCREENRES_800x600 = 4,
  SCREENRES_1024x768 = 5,
  SCREENRES_1280x720 = 6,
  SCREENRES_1280x768 = 7,
  SCREENRES_1280x1024 = 8,
  SCREENRES_1440x900 = 9,
};
```

**File**

VirtualUIS.ridl

**Description**

This is record VirtualUIS::ScreenResolution.

**Library**

Server Configuration Library
SCREENRES_1440x1050 = 10,
SCREENRES_1600x1200 = 11,
SCREENRES_1680x1050 = 12,
SCREENRES_1920x1080 = 13,
SCREENRES_1920x1200 = 14
};

File
VirtualUIS.ridl

Description
This is record VirtualUIS::ScreenResolution.

Library
Server Configuration Library

9.1.2.4 VirtualUIS::ServerSection Enumeration

IDL
[ uuid(D1266C04-7CA5-4705-9486-363BC24CFABC) ]
enum ServerSection {
    SRVSEC_GENERAL = 0,
    SRVSEC_RDS = 1,
    SRVSEC_APPLICATIONS = 2,
    SRVSEC_LICENSES = 3
};

File
VirtualUIS.ridl

Description
This is record VirtualUIS::ServerSection.

Library
Server Configuration Library

9.1.2.5 IServer Interface

Gives you access to server's settings

Class Hierarchy

IDL
9.1.2.5.1 IServer Methods

The methods of the IServer class are listed here.

Interface

IServer Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binding</td>
<td>Returns the TCP/IP binding parameters</td>
</tr>
<tr>
<td>Certificate</td>
<td>Returns the SSL certificate parameters</td>
</tr>
<tr>
<td>RDSAccounts</td>
<td>Contains Remote Desktop Services accounts. VirtualUI makes use of an interactive session. The default setting is to run applications under the console session, but it can be configured to do it under Remote Desktop Services sessions. For the production environment, it is recommended to set VirtualUI to run applications under its own Remote Desktop Services session. This will ensure that the service is available at all times. Alternatively, you can choose to have VirtualUI run the applications under the console session by configuring the Auto Logon feature on your Windows operating system.</td>
</tr>
<tr>
<td>Profiles</td>
<td>Returns the profiles list.</td>
</tr>
<tr>
<td>Load</td>
<td>Loads all the configuration entries and profiles from file. It is automatically called by constructor.</td>
</tr>
<tr>
<td>Save</td>
<td>Saves the entire configuration parameters and profiles.</td>
</tr>
<tr>
<td>HideSection</td>
<td>Hides a configuration section in the VirtualUI Server Manager.</td>
</tr>
<tr>
<td>ShowSection</td>
<td>Makes visible a configuration section in the VirtualUI Server Manager.</td>
</tr>
<tr>
<td>License</td>
<td>Returns the current VirtualUI Server's licence.</td>
</tr>
</tbody>
</table>

9.1.2.5.1.1 IServer::Binding Method

Returns the TCP/IP binding parameters
IDL
[propget, id(0x000000C9)]
HRESULT _stdcall Binding([out, retval] IBinding** Value);

See Also
IBinding interface

Group
IServer Methods

9.1.2.5.1.2 IServer::Certificate Method

Returns the SSL certificate parameters

IDL
[propget, id(0x000000CA)]
HRESULT _stdcall Certificate([out, retval] ICertificate** Value);

See Also
ICertificate interface

Group
IServer Methods

9.1.2.5.1.3 IServer::RDSAccounts Method

Contains Remote Desktop Services accounts. VirtualUI makes use of an interactive session. The default setting is to run applications under the console session, but it can be configured to do it under Remote Desktop Services sessions. For the production environment, it is recommended to set VirtualUI to run applications under its own Remote Desktop Services session. This will ensure that the service is available at all times. Alternatively, you can choose to have VirtualUI run the applications under the console session by configuring the Auto Logon feature on your Windows operating system.

IDL
[propget, id(0x000000CB)]
HRESULT _stdcall RDSAccounts([out, retval] IRDSAccounts** Value);

See Also
IRDSAccounts interface

Group
IServer Methods
9.1.2.5.1.4 IServer::Profiles Method

Returns the profiles list.

**IDL**

[propget, id(0x000000CC)]

HRESULT _stdcall Profiles([out, retval] IProfiles** Value);

≡ **See Also**

IProfiles interface

≡ **Group**

IServer Methods

9.1.2.5.1.5 IServer::Load Method

Loads all the configuration entries and profiles from file. It is automatically called by constructor.

**IDL**

[id(0x000000CD)]

HRESULT _stdcall Load();

≡ **Group**

IServer Methods

9.1.2.5.1.6 IServer::Save Method

Saves the entire configuration parameters and profiles.

**IDL**

[id(0x000000CE)]

HRESULT _stdcall Save();

≡ **Group**

IServer Methods

9.1.2.5.1.7 IServer::HideSection Method

Hides a configuration section in the VirtualUI Server Manager.

**IDL**

[id(0x000000CF)]

HRESULT _stdcall HideSection([in] enum ServerSection section);

≡ **Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2015, Cybele Software, Inc.
### Group

#### IServer Methods

#### 9.1.2.5.1.8 IServer::ShowSection Method

Makes visible a configuration section in the VirtualUI Server Manager.

**IDL**

```
[id(0x000000D0)]
HRESULT _stdcall ShowSection([in] enum ServerSection section);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| [in] enum ServerSection section | The Server configuration section to hide to user. Use one of the following constants:  
- SRVSEC_GENERAL: Hides the General tab, that contains the Binding configuration.  
- SRVSEC_RDS: Hides the tab with the Remote Desktop Services account configuration.  
- SRVSEC_APPLICATIONS: Hides the list of applications.  
- SRVSEC_LICENSES: Hides the tab with License information. |

#### 9.1.2.5.1.9 IServer::License Method

Returns the current VirtualUI Server’s licence.

**IDL**

```
[propget, id(0x000000D1)]
HRESULT _stdcall License([out, retval] ILicense** Value);
```

#### See Also

ILicense interface
Group
IServer Methods

9.1.2.6 ILicense Interface

Contains methods and properties to control VirtualUI Server's licence activation.

Class Hierarchy

IDL
[ uuid(A1DF5DC4-7157-4643-B28F-3B3D20A0E5C8), dual, oleautomation ]
interface ILicense : IDispatch;

File
VirtualUIS.ridl

Library
Server Configuration Library

9.1.2.6.1 ILicense Methods

The methods of the ILicense class are listed here.

Interface
ILicense Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activate</td>
<td>Activates the Server's machine license.</td>
</tr>
<tr>
<td>Deactivate</td>
<td>Deactivates a previously activated license.</td>
</tr>
</tbody>
</table>

9.1.2.6.1.1 ILicense::Activate Method

Activates the Server's machine license.

IDL
[id(0x000000CE)]
HRESULT _stdcall Activate([in] BSTR customerId, [in] BSTR serial, [out] long* resultCode, [out] BSTR* resultText, [out, retval] VARIANT_BOOL* Value);

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2015, Cybele Software, Inc.
<table>
<thead>
<tr>
<th>[in] BSTR customerId</th>
<th>Customer identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR serial</td>
<td>Serial number</td>
</tr>
<tr>
<td>[out] long* resultCode</td>
<td>Result code</td>
</tr>
<tr>
<td>resultText</td>
<td>Error message</td>
</tr>
</tbody>
</table>

✉️ Returns
True if the license was successfully activated. False otherwise (in which case check resultCode and resultText).

✉️ Group
ILicense Methods

9.1.2.6.1.2 ILicense::Deactivate Method
Deactivates a previously activated license.

IDL
[id(0x000000CF)]
HRESULT _stdcall Deactivate();

✉️ Group
ILicense Methods

9.1.2.6.2 Properties

9.1.2.6.2.1 CustomerID
Customer identification.

✉️ Interface
ILicense Interface

9.1.2.6.2.2 Limits
Returns the License limits, if any (ie, trial days, max servers, max users per installation, etc).

✉️ Interface
ILicense Interface

9.1.2.6.2.3 Features
Returns custom features enabled on the License, if any.
9.1.2.6.2.4  IsTrial

Returns true if the current License is in trial mode.

9.1.2.6.2.5  SerialStr

Serial number of the current License.

9.1.2.7  IProfile Interface

A profile contains information about an application or web link configured to be opened in VirtualUI’s home page (or directly through its URL).

Class Hierarchy

IDL
[ uuid(D478CC7A-8071-47BD-BA2D-845131B51B42), dual, oleautomation ]
interface IProfile : IDispatch;

File
VirtualUIS.ridl

Library
Server Configuration Library

9.1.2.7.1  Properties
## Topics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Internal profile ID. This value is auto generated by the library when the profile is created.</td>
</tr>
<tr>
<td>Name</td>
<td>Profile name. It's the caption shown under the application's icon in VirtualUI index page.</td>
</tr>
<tr>
<td>VirtualPath</td>
<td>The Virtual Path unique to this profile.</td>
</tr>
<tr>
<td>IsDefault</td>
<td>This option is used to make this profile the default application.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Enables or disables the profile. Disabled profiles are not accessible by users.</td>
</tr>
<tr>
<td>ProfileKind</td>
<td>Gets or sets the profile type: Application or Web Link. Uses the PROFILE_APP and PROFILE_WEBLINK constants.</td>
</tr>
<tr>
<td>FileName</td>
<td>Complete path of the application executable file. Only used when the ProfileKind is Application.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Parameters to be passed to application.</td>
</tr>
<tr>
<td>StartDir</td>
<td>Application startup directory. In most cases, the same directory of the application executable file.</td>
</tr>
<tr>
<td>UserName</td>
<td>A valid Windows User account to run the application.</td>
</tr>
<tr>
<td>Password</td>
<td>Password of the Windows User account.</td>
</tr>
<tr>
<td>ScreenResolution</td>
<td>Screen resolution in the browser. Uses the constants SCREENRES...</td>
</tr>
<tr>
<td>WebLink</td>
<td>Complete Web Link URL (used only when ProfileKind is Web Link).</td>
</tr>
<tr>
<td>HomePage</td>
<td>Use it to set a customized home page for the application.</td>
</tr>
<tr>
<td>IdleTimeout</td>
<td>Set a timeout in minutes if you want VirtualUI Server to wait before terminating the application once the browser has been closed. Timeout 0 will terminate the application immediately after the browser has been closed.</td>
</tr>
<tr>
<td>IconData</td>
<td>Contains the icon of the profile, consisting in a PNG image in base64 format.</td>
</tr>
</tbody>
</table>

### 9.1.2.7.1.1 ID

Internal profile ID. This value is auto generated by the library when the profile is created.

### Group

#### Properties

### 9.1.2.7.1.2 Name

Profile name. It's the caption shown under the application's icon in VirtualUI index page.
Group Properties

9.1.2.7.1.3 VirtualPath

The Virtual Path unique to this profile.

Group Properties

9.1.2.7.1.4 IsDefault

This option is used to make this profile the default application.

Group Properties

9.1.2.7.1.5 Enabled

Enables or disables the profile. Disabled profiles are not accessible by users.

Group Properties

9.1.2.7.1.6 ProfileKind

Gets or sets the profile type: Application or Web Link. Uses the PROFILE_APP and PROFILE_WEBLINK constants.

Group Properties

9.1.2.7.1.7 FileName

Complete path of the application executable file. Only used when the ProfileKind is Application.
9.1.2.7.1.8 Arguments

Parameters to be passed to application.

- **Group**
  - **Properties**

9.1.2.7.1.9 StartDir

Application startup directory. In most cases, the same directory of the application executable file.

- **Group**
  - **Properties**

9.1.2.7.1.10 UserName

A valid Windows User account to run the application.

- **Group**
  - **Properties**

9.1.2.7.1.11 Password

Password of the Windows User account.

- **Group**
  - **Properties**

9.1.2.7.1.12 ScreenResolution

Screen resolution in the browser. Uses the constants SCREENRES__...

- **Group**
  - **Properties**

9.1.2.7.1.13 WebLink

Complete Web Link URL (used only when ProfileKind is Web Link).

- **Group**
  - **Properties**
9.1.2.7.1.14 HomePage

Use it to set a customized home page for the application.

- Group
  - Properties

9.1.2.7.1.15 IdleTimeout

Set a timeout in minutes if you want VirtualUI Server to wait before terminating the application once the browser has been closed. Timeout 0 will terminate the application immediately after the browser has been closed.

- Group
  - Properties

9.1.2.7.1.16 IconData

Contains the icon of the profile, consisting in a PNG image in base64 format.

- Group
  - Properties

9.1.2.8 IProfiles Interface

Contains the list of profiles registered in VirtualUI Server.

- Class Hierarchy

```
IDL
[ uuid(C271394D-82FA-4DF9-A603-9927AA76A4F9), dual, oleautomation ]
interface IProfiles : IDispatch;
```

- File
  - VirtualUIS.ridl

- Library
  - Server Configuration Library

9.1.2.8.1 IProfiles Methods

The methods of the IProfiles class are listed here.
## Interface

**IProfiles Interface**

## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Creates a new profile and adds it to the list.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes a profile from the list.</td>
</tr>
</tbody>
</table>

### 9.1.2.8.1.1 IProfiles::Add Method

Creates a new profile and adds it to the list.

```IDL
[id(0x000000CD)]
HRESULT _stdcall Add([out, retval] IProfile** Value);
```

### Returns

The newly created profile.

### See Also

**IProfile** interface

### Group

**IProfiles Methods**

### 9.1.2.8.1.2 IProfiles::Delete Method

Deletes a profile from the list.

```IDL
[id(0x000000CE)]
HRESULT _stdcall Delete([in] IProfile* profile);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] IProfile* profile</td>
<td>The profile to be deleted.</td>
</tr>
</tbody>
</table>

### Group

**IProfiles Methods**
9.1.2.8.2 Properties

9.1.2.8.2.1 Count

Returns the profile count.

- **Interface**
  - IPProfiles Interface

9.1.2.8.2.2 Item

Returns a profile from the list by its index.

- **Description**
  - Profile interface.

- **See Also**
  - IProfile

- **Interface**
  - IPProfiles Interface

9.1.2.9 IBinding Interface

Interface for the server’s binding parameters.

- **Class Hierarchy**
  - IDispatch
  - VirtualUIS:IBinding

IDL
[ uuid(52C63E8D-2FA4-4179-AFDB-2D33853F3356), dual, oleautomation ]

interface IBinding : IDispatch;

- **File**
  - VirtualUIS.ridl

- **Library**
  - Server Configuration Library
9.1.2.9.1 Properties

9.1.2.9.1.1 Protocol

Gets o sets the network protocol: HTTP or HTTPS. Uses the PROTO_HTTP and PROTO_HTTPS constants.

interface IBinding Interface

9.1.2.9.1.2 IPAddress

Gets o sets the local IP Address. Leave empty to use all addresses.

interface IBinding Interface

9.1.2.9.1.3 Port

Gets o sets the listening port.

interface IBinding Interface

9.1.2.10 ICertificate Interface

Manages the certificate's configuration for HTTPS Binding.

IDL
[ uuid(8B534446-EDC5-4EE7-91B0-13B5DACC5B51), dual, oleautomation ]
interface ICertificate : IDispatch;

File
VirtualUIS.ridl

Library
Server Configuration Library
9.1.2.10.1 Properties

9.1.2.10.1.1 CertFile

Gets o sets the Certificate file path.

- **Interface**
  - ICertificate Interface

9.1.2.10.1.2 CAFile

Gets o sets the Certificate Authority file path.

- **Interface**
  - ICertificate Interface

9.1.2.10.1.3 PKFile

Gets o sets the path of Private Key file.

- **Interface**
  - ICertificate Interface

9.1.2.10.1.4 PassPhrase

Gets o sets the certificate password.

- **Interface**
  - ICertificate Interface

9.1.2.11 IRDS Interface

Manages the configuration of a Remote Desktop Services account.

- **Class Hierarchy**
  - IDispatch
  - VirtualUIS:IRDS

**IDL**

```idl
[ uuid(103B86C8-E012-4AC7-A366-D3845BBB8D5E), dual, oleautomation ]
interface IRDS : IDispatch;
```

- **File**
  - VirtualUIS.ridl

© 2015, Cybele Software, Inc.
Library

Server Configuration Library

9.1.2.11.1 Properties

9.1.2.11.1.1 Enabled

Enable or disable the use of this RDS account.

Interface

IRDS Interface

9.1.2.11.1.2 UserName

Summary

Gets o sets the RDS Username.

Interface

IRDS Interface

9.1.2.11.1.3 Password

Gets o sets the RDS Password.

Interface

IRDS Interface

9.1.2.12 Server CoClass

Class Hierarchy

VirtualUIS::Server

IDL

[ uuid(67F012A8-5C8D-4E30-B697-104AF434CF57) ]
coclass Server;

File

VirtualUIS.ridl

Description

This is class VirtualUIS::Server.
9.1.2.13  VirtualUIS::PWideString1 Type

IDL

typedef BSTR* PWideString1;

9.1.2.14  IRDSAccounts Interface

Contains a list of Remote Desktop Services accounts.

Class Hierarchy

IDL

interface IRDSAccounts : IDispatch;

9.1.2.14.1  IRDSAccounts Methods

The methods of the IRDSAccounts class are listed here.
## Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>This is Count, a member of class IRDSAccounts.</td>
</tr>
<tr>
<td>Item</td>
<td>This is Item, a member of class IRDSAccounts.</td>
</tr>
<tr>
<td>Add</td>
<td>This is Add, a member of class IRDSAccounts.</td>
</tr>
<tr>
<td>Delete</td>
<td>This is Delete, a member of class IRDSAccounts.</td>
</tr>
</tbody>
</table>

### 9.1.2.14.1.1 IRDSAccounts::Count Method

**IDL**

```idl
[propget, id(0x000000C9)]
HRESULT _stdcall Count([out, retval] long* Value);
```

**Description**

This is Count, a member of class IRDSAccounts.

**Group**

IRDSAccounts Methods

### 9.1.2.14.1.2 IRDSAccounts::Item Method

**IDL**

```idl
[propget, id(0x000000CA), defaultcollelem]
HRESULT _stdcall Item([in] long index, [out, retval] IRDS** Value);
```

**Description**

This is Item, a member of class IRDSAccounts.

**Group**

IRDSAccounts Methods

### 9.1.2.14.1.3 IRDSAccounts::Add Method

**IDL**

```idl
[id(0x000000CB)]
HRESULT _stdcall Add([in] BSTR UserName, [in] BSTR Password, [in, defaultvalue(False)] VARIANT_BOOL CreateAccount, [out, retval] IRDS** Value);
```

**Description**

This is Add, a member of class IRDSAccounts.

**Group**

IRDSAccounts Methods
IRDSAccounts Methods

9.1.2.14.1.4 IRDSAccounts::Delete Method

**IDL**

```
[id(0x000000CC)]
HRESULT _stdcall Delete([in] BSTR UserName, [in] VARIANT_BOOL DeleteAccount, [out, retval] VARIANT_BOOL* Value);
```

**Description**

This is Delete, a member of class IRDSAccounts.

**Group**

IRDSAccounts Methods

9.2 .NET Classes

**Namespaces**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cybele.Thinfinity</td>
<td>This is namespace Cybele.Thinfinity.</td>
</tr>
<tr>
<td>Cybele.Thinfinity.Settings.VirtualUI</td>
<td>This is namespace Cybele.Thinfinity.Settings.VirtualUI.</td>
</tr>
</tbody>
</table>

9.2.1 Cybele.Thinfinity Namespace

This is namespace Cybele.Thinfinity.

**Classes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientSettings</td>
<td>Allows to set some client settings.</td>
</tr>
<tr>
<td>BrowserInfo</td>
<td>Contains information regarding the end-user's screen, web browser, the window containing VirtualUI Viewer and VirtualUI Viewer itself. The VirtualUI Viewer runs inside an HTML DIV element contained in a frame of a browser window on the end-user's application page.</td>
</tr>
<tr>
<td>DevServer</td>
<td>Contains properties to manage the VirtualUI Development Server as well as the access from the developer's web browser.</td>
</tr>
<tr>
<td>BrowserResizeEventArgs</td>
<td>This is class Cybele.Thinfinity.BrowserResizeEventArgs.</td>
</tr>
<tr>
<td>GetUploadDirEventArgs</td>
<td>This is class Cybele.Thinfinity.GetUploadDirEventArgs.</td>
</tr>
<tr>
<td>DownloadEndArgs</td>
<td>This is class Cybele.Thinfinity.DownloadEndArgs.</td>
</tr>
<tr>
<td>CloseArgs</td>
<td>This is class Cybele.Thinfinity.CloseArgs.</td>
</tr>
<tr>
<td>ReceiveMessageArgs</td>
<td>This is class Cybele.Thinfinity.ReceiveMessageArgs.</td>
</tr>
</tbody>
</table>
VirtualUI

Main class. Has methods, properties and events to allow the activation and control the behavior of VirtualUI.

JSExecuteMethodEventArgs

This is class Cybele.Thinfinity.JSExecuteMethodEventArgs.

JSPROPERTYCHANGEEventArgs

This is class Cybele.Thinfinity.JSPROPERTYCHANGEEventArgs.

JSObject

Represents a custom remotable object.

JSBinding

This is class Cybele.Thinfinity.JSBinding.

JSCallback

This is class Cybele.Thinfinity.JSCallback.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrowserOrientation</td>
<td>This is record Cybele.Thinfinity.BrowserOrientation.</td>
</tr>
<tr>
<td>MouseMoveGestureStyle</td>
<td>This is record Cybele.Thinfinity.MouseMoveGestureStyle.</td>
</tr>
<tr>
<td>MouseMoveGestureAction</td>
<td>This is record Cybele.Thinfinity.MouseMoveGestureAction.</td>
</tr>
</tbody>
</table>

```
C#
public enum BrowserOrientation {
    PORTRAIT = 0,
    LANDSCAPE = 1
}
```

File

Thinfinity.VirtualUI.cs

Description

This is record Cybele.Thinfinity.BrowserOrientation.

Namespace

Cybele.Thinfinity Namespace
9.2.1.2 Cybele.Thinfinity.MouseMoveGestureStyle Enumeration

C#

```csharp
public enum MouseMoveGestureStyle {
    MM_STYLE_RELATIVE = 0,
    MM_STYLE_ABSOLUTE = 1
}
```

File

Thinfinity.VirtualUI.cs

Description

This is record Cybele.Thinfinity.MouseMoveGestureStyle.

Namespace

Cybele.Thinfinity Namespace

9.2.1.3 Cybele.Thinfinity.MouseMoveGestureAction Enumeration

C#

```csharp
public enum MouseMoveGestureAction {
    MM_ACTION_MOVE = 0,
    MM_ACTION_WHEEL = 1
}
```

File

Thinfinity.VirtualUI.cs

Description

This is record Cybele.Thinfinity.MouseMoveGestureAction.

Namespace

Cybele.Thinfinity Namespace

9.2.1.4 ClientSettings Class

Allows to set some client settings.

Class Hierarchy

```
IClientSettings
+ Cybele.Thinfinity.ClientSettings
+ IDisposable
```

C#
public class ClientSettings : IClientSettings, IDisposable;

- **File**
  Thinfinity.VirtualUI.cs

- **Namespace**
  Cybele.Thinfinity Namespace

### 9.2.1.4.1 ClientSettings.ClientSettings Constructor

**C#**
```csharp
public ClientSettings (IVirtualUI virtualUI);
```

- **Description**
  This is ClientSettings, a member of class ClientSettings.

- **Class**
  ClientSettings Class

### 9.2.1.4.2 ClientSettings Methods

The methods of the ClientSettings class are listed here.

- **Class**
  ClientSettings Class

#### 9.2.1.4.2.1 ClientSettings.Dispose Method

**C#**
```csharp
public void Dispose();
```

- **Description**
  This is Dispose, a member of class ClientSettings.

- **Group**
  ClientSettings Methods
9.2.1.4.3 ClientSettings Properties

The properties of the ClientSettings class are listed here.

## Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MouseMoveGestureStyle</td>
<td>Valid for touch devices. Specifies whether the mouse pointer is shown and acts on the exact spot of the finger touch (absolute) or its position is managed relatively to the movement of the finger touch (relative).</td>
</tr>
<tr>
<td>MouseMoveGestureAction</td>
<td>Specifies whether the &quot;mouse move&quot; simulation on a touch device is interpreted as a mouse move or as a mouse wheel.</td>
</tr>
<tr>
<td>CursorVisible</td>
<td>Hides/shows the mouse pointer.</td>
</tr>
</tbody>
</table>

### 9.2.1.4.3.1 ClientSettings.MouseMoveGestureStyle Property

Valid for touch devices. Specifies whether the mouse pointer is shown and acts on the exact spot of the finger touch (absolute) or its position is managed relatively to the movement of the finger touch (relative).

**C#**

```csharp
public MouseMoveGestureStyle MouseMoveGestureStyle;
```

### 9.2.1.4.3.2 ClientSettings.MouseMoveGestureAction Property

Specifies whether the "mouse move" simulation on a touch device is interpreted as a mouse move or as a mouse wheel.

**C#**

```csharp
public MouseMoveGestureAction MouseMoveGestureAction;
```

### 9.2.1.4.3.3 ClientSettings.CursorVisible Property

Hides/shows the mouse pointer.

**C#**

```csharp
public CursorVisible CursorVisible;
```
public bool CursorVisible;

--- Group

ClientSettings Properties

9.2.1.5 BrowserInfo Class

Contains information regarding the end-user's screen, web browser, the window containing VirtualUI Viewer and VirtualUI Viewer itself. The VirtualUI Viewer runs inside an HTML DIV element contained in a frame of browser window on the end-user's application page.

--- Class Hierarchy

C#
public class BrowserInfo : IBrowserInfo, IDisposable;

--- File

Thinfinity.VirtualUI.cs

--- Namespace

Cybele.Thinfinity Namespace

9.2.1.5.1 BrowserInfo.BrowserInfo Constructor

C#
public BrowserInfo(IVirtualUI virtualUI);

--- Description

This is BrowserInfo, a member of class BrowserInfo.

--- Class

BrowserInfo Class

9.2.1.5.2 BrowserInfo Methods

The methods of the BrowserInfo class are listed here.

--- Class

BrowserInfo Class

--- Public Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>This is Dispose, a member of class BrowserInfo.</td>
</tr>
<tr>
<td>GetCookie</td>
<td>Returns a browser's cookie value.</td>
</tr>
<tr>
<td>SetCookie</td>
<td>Sets a cookie in the browser.</td>
</tr>
</tbody>
</table>

### 9.2.1.5.2.1 BrowserInfo.Dispose Method

**C#**

```csharp
public void Dispose();
```

**Description**

This is Dispose, a member of class BrowserInfo.

**Group**

BrowserInfo Methods

### 9.2.1.5.2.2 BrowserInfo.GetCookie Method

Returs a browser's cookie value.

**C#**

```csharp
public string GetCookie(string Name);
```

**Group**

BrowserInfo Methods

### 9.2.1.5.2.3 BrowserInfo.SetCookie Method

Sets a cookie in the browser.

**C#**

```csharp
public void SetCookie(string Name, string Value, string Expires);
```

**Group**

BrowserInfo Methods

### 9.2.1.5.3 BrowserInfo Properties

The properties of the BrowserInfo class are listed here.

**Class**

BrowserInfo Class

© 2015, Cybele Software, Inc.
### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewWidth</td>
<td>Returns the width of the VirtualUI Viewer.</td>
</tr>
<tr>
<td>ViewHeight</td>
<td>Returns the height of the VirtualUI Viewer.</td>
</tr>
<tr>
<td>BrowserWidth</td>
<td>Returns the width of the HTML element containing the VirtualUI Viewer.</td>
</tr>
<tr>
<td>BrowserHeight</td>
<td>Returns the height of the HTML element containing the VirtualUI Viewer.</td>
</tr>
<tr>
<td>ScreenWidth</td>
<td>Returns the width of the end-user's monitor screen.</td>
</tr>
<tr>
<td>ScreenHeight</td>
<td>Returns the height of the end-user's monitor screen.</td>
</tr>
<tr>
<td>Username</td>
<td>Returns the logged-on Username.</td>
</tr>
<tr>
<td>IPAddress</td>
<td>Returns the client's IP address.</td>
</tr>
<tr>
<td>UserAgent</td>
<td>Returns the browser's User Agent string.</td>
</tr>
<tr>
<td>UniqueBrowserId</td>
<td>UniqueBrowserId identifies an instance of a Web Browser. Each time an end-user opens the application from a different browser window, this ID will have a different value.</td>
</tr>
<tr>
<td>Location</td>
<td>Returns the URL of the current application.</td>
</tr>
<tr>
<td>ScreenResolution</td>
<td>Returns the application screen resolution defined in the application profile.</td>
</tr>
<tr>
<td>Orientation</td>
<td>Returns the browser's orientation.</td>
</tr>
</tbody>
</table>

#### 9.2.1.5.3.1 BrowserInfo.ViewWidth Property

Returns the width of the VirtualUI Viewer.

**C#**
```
public int ViewWidth;
```

#### Group

BrowserInfo Properties

#### 9.2.1.5.3.2 BrowserInfo.ViewHeight Property

Returns the height of the VirtualUI Viewer.

**C#**
```
public int ViewHeight;
```

#### Group
9.2.1.5.3.3 BrowserInfo.BrowserWidth Property

```csharp
public int BrowserWidth;
```

**Description**

Returns the width of the HTML element containing the VirtualUI Viewer.

**Group**

BrowserInfo Properties

9.2.1.5.3.4 BrowserInfo.BrowserHeight Property

Returns the height of the HTML element containing the VirtualUI Viewer.

```csharp
public int BrowserHeight;
```

**Group**

BrowserInfo Properties

9.2.1.5.3.5 BrowserInfo.ScreenWidth Property

Returns the width of the end-user's monitor screen.

```csharp
public int ScreenWidth;
```

**Group**

BrowserInfo Properties

9.2.1.5.3.6 BrowserInfo.ScreenHeight Property

Returns the height of the end-user's monitor screen.

```csharp
public int ScreenHeight;
```

**Group**

BrowserInfo Properties
9.2.1.5.3.7 BrowserInfo.Username Property

Returns the logged-on Username.

```
C#
public string Username;
```

9.2.1.5.3.8 BrowserInfo.IPAddress Property

Returns the client's IP address.

```
C#
public string IPAddress;
```

9.2.1.5.3.9 BrowserInfo.UserAgent Property

Returns the browser’s User Agent string.

```
C#
public string UserAgent;
```

9.2.1.5.3.10 BrowserInfo.UniqueBrowserId Property

UniqueBrowserId identifies an instance of a Web Browser. Each time an end-user opens the application from a different browser window, this ID will have a different value.

```
C#
public string UniqueBrowserId;
```

9.2.1.5.3.11 BrowserInfo.Location Property

Returns the URL of the current application.

```
C#
public string Location;
```
Group

BrowserInfo Properties

9.2.1.5.3.12 BrowserInfo.ScreenResolution Property

Returns the application screen resolution defined in the application profile.

C# public int ScreenResolution;

Group

BrowserInfo Properties

9.2.1.5.3.13 BrowserInfo.Orientation Property

Returns the browser's orientation.

C# public BrowserOrientation Orientation;

Group

BrowserInfo Properties

9.2.1.6 DevServer Class

Contains properties to manage the VirtualUI Development Server as well as the access from the developer's web browser.

Class Hierarchy

C# public class DevServer : IDevServer, IDisposable;

File

Thinfinity.VirtualUI.cs

Namespace

Cybele.Thinfinity Namespace
9.2.1.6.1 DevServer.DevServer Constructor

C#

```csharp
public DevServer(IVirtualUI virtualUI);
```

**Description**

This is DevServer, a member of class DevServer.

**Class**

DevServer Class

9.2.1.6.2 DevServer Methods

The methods of the DevServer class are listed here.

**Class**

DevServer Class

**Public Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>This is Dispose, a member of class DevServer.</td>
</tr>
</tbody>
</table>

9.2.1.6.2.1 DevServer.Dispose Method

C#

```csharp
public void Dispose();
```

**Description**

This is Dispose, a member of class DevServer.

**Group**

DevServer Methods

9.2.1.6.3 DevServer Properties

The properties of the DevServer class are listed here.

**Class**

DevServer Class

**Public Properties**
### Name | Description
--- | ---
Enabled | Enables/disables the Development Server.
Port | Gets/sets the Development Server’s TCP/IP listening port.
StartBrowser | Instructs VirtualUI whether start or not the local web browser upon VirtualUI activation.

#### 9.2.1.6.3.1 DevServer.Enabled Property

Enables/disables the Development Server.

**C#**
```
public bool Enabled;
```

**Group**
DevServer Properties

#### 9.2.1.6.3.2 DevServer.Port Property

Gets/sets the Development Server’s TCP/IP listening port.

**C#**
```
public int Port;
```

**Group**
DevServer Properties

#### 9.2.1.6.3.3 DevServer.StartBrowser Property

Instructs VirtualUI whether start or not the local web browser upon VirtualUI activation.

**C#**
```
public bool StartBrowser;
```

**Group**
DevServer Properties

#### 9.2.1.7 BrowserResizeEventArgs Class

**Class Hierarchy**
```
EventArgs  
| Cybele.Thinfinity.BrowserResizeEventArgs
```

**C#**
```
public class BrowserResizeEventArgs : EventArgs;
```

**File**
Description
This is class Cybele.Thinfinity.BrowserResizeEventArgs.

Namespace
Cybele.Thinfinity Namespace

9.2.1.7.1 BrowserResizeEventArgs Properties
The properties of the BrowserResizeEventArgs class are listed here.

Class
BrowserResizeEventArgs Class

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>This is Width, a member of class BrowserResizeEventArgs.</td>
</tr>
<tr>
<td>Height</td>
<td>This is Height, a member of class BrowserResizeEventArgs.</td>
</tr>
<tr>
<td>ResizeMaximized</td>
<td>This is ResizeMaximized, a member of class BrowserResizeEventArgs.</td>
</tr>
</tbody>
</table>

9.2.1.7.1.1 BrowserResizeEventArgs.Width Property

C#
```csharp
public int Width;
```

Description
This is Width, a member of class BrowserResizeEventArgs.

Group
BrowserResizeEventArgs Properties

9.2.1.7.1.2 BrowserResizeEventArgs.Height Property

C#
```csharp
public int Height;
```

Description
This is Height, a member of class BrowserResizeEventArgs.
9.2.1.7.1.3 BrowserResizeEventArgs.ResizeMaximized Property

**C#**

```csharp
public bool ResizeMaximized;
```

**Description**

This is ResizeMaximized, a member of class BrowserResizeEventArgs.

9.2.1.8 GetUploadDirEventArgs Class

**Class Hierarchy**

```
EventArgs -> Cybele.Thinfinity.GetUploadDirEventArgs
```

**C#**

```csharp
public class GetUploadDirEventArgs : EventArgs;
```

**File**

Thinfinity.VirtualUI.cs

**Description**

This is class Cybele.Thinfinity.GetUploadDirEventArgs.

**Namespace**

Cybele.Thinfinity Namespace

9.2.1.8.1 GetUploadDirEventArgs Properties

The properties of the GetUploadDirEventArgs class are listed here.

**Class**

GetUploadDirEventArgs Class

**Public Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
</tbody>
</table>

© 2015, Cybele Software, Inc.
### 9.2.1.8.1.1 GetUploadDirEventArgs.Directory Property

```csharp
public string Directory;
```

**Description**

This is Directory, a member of class GetUploadDirEventArgs.

**Group**

GetUploadDirEventArgs Properties

### 9.2.1.8.1.2 GetUploadDirEventArgs.Handled Property

```csharp
public bool Handled;
```

**Description**

This is Handled, a member of class GetUploadDirEventArgs.

**Group**

GetUploadDirEventArgs Properties

### 9.2.1.9 DownloadEndArgs Class

**Class Hierarchy**

C#

```csharp
public class DownloadEndArgs : EventArgs;
```

**File**

Thinfinity.VirtualUI.cs

**Description**

This is class Cybele.Thinfinity.DownloadEndArgs.

**Namespace**

Cybele.Thinfinity Namespace
9.2.1.9.1 DownloadEndArgs Properties

The properties of the DownloadEndArgs class are listed here.

- **Class**
  - DownloadEndArgs Class

### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>This is Filename, a member of class DownloadEndArgs.</td>
</tr>
</tbody>
</table>

9.2.1.9.1.1 DownloadEndArgs.Filename Property

- **C#**
  ```csharp
  public string Filename;
  ```

- **Description**
  This is Filename, a member of class DownloadEndArgs.

- **Group**
  - DownloadEndArgs Properties

9.2.1.10 CloseArgs Class

- **Class Hierarchy**
  ```
  public class CloseArgs : EventArgs;
  ```

- **File**
  Thinfinity.VirtualUI.cs

- **Description**
  This is class Cybele.Thinfinity.CloseArgs.

- **Namespace**
  - Cybele.Thinfinity Namespace
9.2.1.11 ReceiveMessageArgs Class

Class Hierarchy
public class ReceiveMessageArgs : EventArgs;

File
Thinfinity.VirtualUI.cs

Description
This is class Cybele.Thinfinity.ReceiveMessageArgs.

Namespace
Cybele.Thinfinity Namespace

9.2.1.11.1 ReceiveMessageArgs Properties

The properties of the ReceiveMessageArgs class are listed here.

Class
ReceiveMessageArgs Class

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>This is Data, a member of class ReceiveMessageArgs.</td>
</tr>
</tbody>
</table>

9.2.1.11.1.1 ReceiveMessageArgs.Data Property

C#

```csharp
public string Data;
```

Description
This is Data, a member of class ReceiveMessageArgs.

Group
ReceiveMessageArgs Properties
9.2.1.12 VirtualUI Class

Main class. Has methods, properties and events to allow the activation and control the behavior of VirtualUI.

C#

`public class VirtualUI : VirtualUILibrary, IVirtualUI, IDisposable;`

File

`Thinfinity.VirtualUI.cs`

Namespace

`Cybele.Thinfinity Namespace`

9.2.1.12.1 VirtualUI.VirtualUI Constructor

C#

`public VirtualUI();`

Description

This is VirtualUI, a member of class VirtualUI.

Class

`VirtualUI Class`

9.2.1.12.2 VirtualUI Methods

The methods of the VirtualUI class are listed here.

Class

`VirtualUI Class`

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DownloadFile</td>
<td>This is the overview for the DownloadFile method overload.</td>
</tr>
<tr>
<td>Start</td>
<td>This is the overview for the Start method overload.</td>
</tr>
</tbody>
</table>
9.2.1.12.2.1 DownloadFile Method

This is the overview for the DownloadFile method overload.

<table>
<thead>
<tr>
<th>Overload List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>VirtualUI.DownloadFile (string)</td>
</tr>
<tr>
<td>VirtualUI.DownloadFile (string, string)</td>
</tr>
<tr>
<td>VirtualUI.DownloadFile (string, string, string)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualUI Methods</td>
</tr>
</tbody>
</table>

Sends the specified file to the end-user's web browser for saving it in the remote machine.

```c#
public void DownloadFile(string LocalFilename);
```

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>string LocalFilename</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>DownloadFile Method</td>
</tr>
</tbody>
</table>
Sends the specified file to the end-user's web browser for saving it in the remote machine.

C#
public void DownloadFile(string LocalFilename, string RemoteFilename);

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string LocalFilename</td>
<td>Name of the local file to be sent.</td>
</tr>
<tr>
<td>string RemoteFilename</td>
<td>Name of the file in the remote machine.</td>
</tr>
</tbody>
</table>

#### Group

DownloadFile Method

Sends the specified file to the end-user's web browser for saving it in the remote machine.

C#
public void DownloadFile(string LocalFilename, string RemoteFilename, string MimeType);

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string LocalFilename</td>
<td>Name of the local file to be sent.</td>
</tr>
<tr>
<td>string RemoteFilename</td>
<td>Name of the file in the remote machine.</td>
</tr>
<tr>
<td>string MimeType</td>
<td>content-type of the file. If specified, the content will be handled by browser. Leave blank to force download.</td>
</tr>
</tbody>
</table>

#### Group

DownloadFile Method

9.2.12.12.2 Start Method

This is the overview for the Start method overload.

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualUI.Start ()</td>
<td>Starts the VirtualUI's activation process. Returns true if VirtualUI was fully activated or false if the timeout expired. The timeout is 60 seconds.</td>
</tr>
<tr>
<td>VirtualUI.Start (int)</td>
<td>Starts the VirtualUI's activation process. Returns true if VirtualUI was fully activated or false if the passed timeout expired.</td>
</tr>
</tbody>
</table>

#### Group
VirtualUI Methods

Starts the VirtualUI's activation process. Returns true if VirtualUI was fully activated or false if the timeout expired. The timeout is 60 seconds.

C#
public bool Start();

Group
Start Method

Starts the VirtualUI's activation process. Returns true if VirtualUI was fully activated or false if the passed timeout expired.

C#
public bool Start(int timeout);

Group
Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>Maximum time, in seconds, until the activation process is canceled. Defaults to 60 seconds.</td>
</tr>
</tbody>
</table>

Remarks

To fully activate VirtualUI, the connection with the end-user's web browser must established within the time specified by Timeout parameter.

Group

9.2.1.12.2.3 UploadFile Method

This is the overview for the UploadFile method overload.

Group

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualUI.UploadFile()</td>
<td>Selects a file from client machine, and it's uploaded to VirtualUI public path.</td>
</tr>
<tr>
<td>VirtualUI.UploadFile(string)</td>
<td>Selects a file from client machine, and it's uploaded to ServerDirectory</td>
</tr>
</tbody>
</table>

Group

VirtualUI Methods
Selects a file from client machine, and it's uploaded to ServerDirectory

```csharp
public void UploadFile(string ServerDirectory);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string ServerDirectory</td>
<td>Destination directory in Server.</td>
</tr>
</tbody>
</table>

### Group

#### UploadFile Method

Selects a file from client machine, and it's uploaded to VirtualUI public path.

```csharp
public void UploadFile();
```

### Group

#### UploadFile Method

#### 9.2.1.12.2.4 VirtualUI.Dispose Method

```csharp
public void Dispose();
```

### Description

This is Dispose, a member of class VirtualUI.

### Group

#### VirtualUI Methods

#### 9.2.1.12.2.5 VirtualUI.Stop Method

Deactivates VirtualUI, closing the connection with the end-user's web browser.

```csharp
public void Stop();
```

### Group

#### VirtualUI Methods
9.2.1.12.2.6 VirtualUI.PrintPdf Method

Sends the specified PDF file to be shown on the end-user’s web browser.

**C#**

```csharp
public void PrintPdf(string FileName);
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName</td>
<td>Name of the PDF file.</td>
</tr>
</tbody>
</table>

**Remarks**

PrintPDF is similar to DownloadFile, except that it downloads the file with a content-type: application/pdf.

**Group**

VirtualUI Methods

9.2.1.12.2.7 VirtualUI.OpenLinkDlg Method

Displays a popup with a button to open a website link.

**C#**

```csharp
public void OpenLinkDlg(string Url, string Caption);
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Url</td>
<td>Link to open.</td>
</tr>
<tr>
<td>Caption</td>
<td>Text to display in popup.</td>
</tr>
</tbody>
</table>

**Group**

VirtualUI Methods

9.2.1.12.2.8 VirtualUI.SendMessage Method

Sends a data string to the web browser.

**C#**

```csharp
public void SendMessage(string Data);
```

**Remarks**

This method is used to send custom data to the browser for custom purposes.

**Group**
VirtualUI Methods

9.2.1.12.2.9  VirtualUI.AllowExecute Method

Allows the execution of the passed application.

C#

```csharp
public void AllowExecute(string Filename);
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string Filename</td>
<td>regular expression specifying the filename(s) of the applications allowed to run.</td>
</tr>
</tbody>
</table>

Remarks

Under VirtualUI environment, only applications precompiled with VirtualUI SDK should be allowed to run. Applications not under VirtualUI control, cannot be controlled.

Group

VirtualUI Methods

9.2.1.12.2.10  VirtualUI.SetImageQualityByWnd Method

Allows to the the image quality for the specified window.

C#

```csharp
public void SetImageQualityByWnd(long Wnd, string Classname, int Quality);
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long Wnd</td>
<td>Window handle.</td>
</tr>
<tr>
<td>int Quality</td>
<td>Quality from 0 to 100.</td>
</tr>
<tr>
<td>Class</td>
<td>Window classname.</td>
</tr>
</tbody>
</table>

Group

VirtualUI Methods

9.2.1.12.2.11  VirtualUI.TakeScreenshot Method

Takes a screenshot of a Window.

C#

```csharp
public bool TakeScreenshot(long Wnd, string FileName);
```
### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long Wnd</td>
<td>The Window to capture.</td>
</tr>
<tr>
<td>string FileName</td>
<td>Full path of file to save screenshot. Extensions allowed: jpg, bmp, png.</td>
</tr>
</tbody>
</table>

### Group

#### VirtualUI Methods

#### 9.2.1.12.3 VirtualUI Properties

The properties of the VirtualUI class are listed here.

#### Class

**VirtualUI Class**

#### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Returns the VirtualUI’s state.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Enables/disables VirtualUI for the container application.</td>
</tr>
<tr>
<td>DevMode</td>
<td>Gets/sets the development mode.</td>
</tr>
<tr>
<td>StdDialogs</td>
<td>Enables/disables the use of standard dialogs.</td>
</tr>
<tr>
<td>BrowserInfo</td>
<td>Contains information regarding the end-user’s environment.</td>
</tr>
<tr>
<td>DevServer</td>
<td>Allows for managing the Development Server.</td>
</tr>
<tr>
<td>ClientSettings</td>
<td>Controls some working parameters on the client side.</td>
</tr>
</tbody>
</table>

#### 9.2.1.12.3.1 VirtualUI.Active Property

Returns the VirtualUI’s state.

#### C#

```csharp
public bool Active;
```

### Group

**VirtualUI Properties**
9.2.1.12.3.2 VirtualUI.Enabled Property

Enables/disables `VirtualUI` for the container application.

```csharp
public bool Enabled;
```

Remarks

When in development mode, applications executed under the IDE, connect to the Development Server, allowing the access to the application from the browser while in debugging.

9.2.1.12.3.3 VirtualUI.DevMode Property

Gets/sets the development mode.

```csharp
public bool DevMode;
```

9.2.1.12.3.4 VirtualUI.StdDialogs Property

Enables/disables the use of standard dialogs.

```csharp
public bool StdDialogs;
```

Remarks

When set to false, the standard save, open and print dialogs are replaced by native browser ones, enabling you to extend the operations to the remote computer.

9.2.1.12.3.5 VirtualUI.BrowserInfo Property

Contains information regarding the end-user's environment.

```csharp
public IBrowserInfo BrowserInfo;
```
9.2.1.12.3.6  VirtualUI.DevServer Property

Allows for managing the Development Server.

C#

```csharp
public IDevServer DevServer;
```

9.2.1.12.3.7  VirtualUI.ClientSettings Property

Controls some working parameters on the client side.

C#

```csharp
public IClientSettings ClientSettings;
```

9.2.1.12.4  VirtualUI Events

The events of the VirtualUI class are listed here.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnBrowserResize</td>
<td>This is OnBrowserResize, a member of class VirtualUI.</td>
</tr>
<tr>
<td>OnGetUploadDir</td>
<td>This is OnGetUploadDir, a member of class VirtualUI.</td>
</tr>
<tr>
<td>OnClose</td>
<td>This is OnClose, a member of class VirtualUI.</td>
</tr>
<tr>
<td>OnReceiveMessage</td>
<td>This is OnReceiveMessage, a member of class VirtualUI.</td>
</tr>
<tr>
<td>OnDownloadEnd</td>
<td>This is OnDownloadEnd, a member of class VirtualUI.</td>
</tr>
</tbody>
</table>
9.2.1.12.4.1 VirtualUI.OnBrowserResize Event

C#

```csharp
public event EventHandler<BrowserResizeEventArgs> OnBrowserResize;
```

**Description**
This is OnBrowserResize, a member of class VirtualUI.

**Group**
VirtualUI Events

9.2.1.12.4.2 VirtualUI.OnGetUploadDir Event

C#

```csharp
public event EventHandler<GetUploadDirEventArgs> OnGetUploadDir;
```

**Description**
This is OnGetUploadDir, a member of class VirtualUI.

**Group**
VirtualUI Events

9.2.1.12.4.3 VirtualUI.OnClose Event

C#

```csharp
public event EventHandler<CloseArgs> OnClose;
```

**Description**
This is OnClose, a member of class VirtualUI.

**Group**
VirtualUI Events

9.2.1.12.4.4 VirtualUI.OnReceiveMessage Event

C#

```csharp
public event EventHandler<ReceiveMessageArgs> OnReceiveMessage;
```

**Description**
This is OnReceiveMessage, a member of class VirtualUI.

**Group**
VirtualUI Events

9.2.1.12.4.5 VirtualUI.OnDownloadEnd Event

C#  
```csharp
public event EventHandler<DownloadEndArgs> OnDownloadEnd;
```

■ Description  
This is OnDownloadEnd, a member of class VirtualUI.

■ Group  
VirtualUI Events

9.2.1.13 JSExecuteMethodEventArgs Class

■ Class Hierarchy  
```
EventArgs   
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cybele.Thinfinity.JSExecuteMethodEventArgs</td>
</tr>
</tbody>
</table>
```

C#  
```csharp
public class JSExecuteMethodEventArgs : EventArgs;
```

■ File  
Thinfinity.VirtualUI.cs

■ Description  
This is class Cybele.Thinfinity.JSExecuteMethodEventArgs.

■ Namespace  
Cybele.Thinfinity Namespace

9.2.1.13.1 JSExecuteMethodEventArgs Properties

The properties of the JSExecuteMethodEventArgs class are listed here.

■ Class  
JSExecuteMethodEventArgs Class

■ Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender</td>
<td>This is Sender, a member of class JSExecuteMethodEventArgs.</td>
</tr>
</tbody>
</table>
9.2.13.1.1  JSExecuteMethodEventArgs.Sender Property

C#
public IJSObject Sender;

- Description
This is Sender, a member of class JSExecuteMethodEventArgs.

- Group
JSExecuteMethodEventArgs Properties

9.2.13.1.2  JSExecuteMethodEventArgs.Method Property

C#
public IJSMethod Method;

- Description
This is Method, a member of class JSExecuteMethodEventArgs.

- Group
JSExecuteMethodEventArgs Properties

9.2.14  JSPropertyChangeEventArgs Class

- Class Hierarchy

C#
public class JSPropertyChangeEventArgs : EventArgs;

- File
Thinfinity.VirtualUI.cs

- Description
This is class Cybele.Thinfinity.JSPropertyChangeEventArgs.

- Namespace
Cybele.Thinfinity Namespace
9.2.1.14.1  JSPropertyChangeEventArgs Properties

The properties of the JSPropertyChangeEventArgs class are listed here.

Class
JSPropertyChangeEventArgs Class

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender</td>
<td>This is Sender, a member of class JSPropertyChangeEventArgs.</td>
</tr>
<tr>
<td>Prop</td>
<td>This is Prop, a member of class JSPropertyChangeEventArgs.</td>
</tr>
</tbody>
</table>

9.2.1.14.1.1  JSPropertyChangeEventArgs(Sender Property

```
C#
public IJSObject Sender;
```

Description
This is Sender, a member of class JSPropertyChangeEventArgs.

Group
JSPropertyChangeEventArgs Properties

9.2.1.14.1.2  JSPropertyChangeEventArgs.Prop Property

```
C#
public IJSProperty Prop;
```

Description
This is Prop, a member of class JSPropertyChangeEventArgs.

Group
JSPropertyChangeEventArgs Properties

9.2.1.15  JSObject Class

Represents a custom remotable object.

Class Hierarchy
C#

public class JSObject : VirtualUILibrary, IJSObject, IDisposable;

Remarks

JSObject allows you to define an object model that is mirrored on the client side, and allows for an easy, powerful and straight-forward way to connect the web browser client application and the remoted Windows application.

JSObject can contain properties (IJSProperties), methods (IJSMethods), events (IJSEvents) and children objects. Changes to properties values are propagated in from server to client and viceversa, keeping the data synchronized.

JSObject is defined as a model seen from the client perspective. A method (IJSMethod) is called on the client side and executed on the server side. An event (IJSEvent) is called on the server side and raised on the client side.

Namespace

Cybele.Thinfinity Namespace

9.2.1.15.1 JSObject.JSObject Constructor

C#

public JSObject(string Id);

Description

This is JSObject, a member of class JSObject.

Class

JSObject Class

9.2.1.15.2 JSObject Methods

The methods of the JSObject class are listed here.

Class

JSObject Class
### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>This is Dispose, a member of class JSObject.</td>
</tr>
<tr>
<td>FireEvent</td>
<td>This is FireEvent, a member of class JSObject.</td>
</tr>
<tr>
<td>ApplyChanges</td>
<td>When this method called, all properties getters are internally called looking for changes. Any change to the property value is sent to the client.</td>
</tr>
<tr>
<td>ApplyModel</td>
<td>Propagates the whole JSObject definition to the javascript client.</td>
</tr>
</tbody>
</table>

#### 9.2.1.15.2.1  JSObject.Dispose Method

```csharp
public void Dispose();
```

**Description**

This is Dispose, a member of class JSObject.

**Group**

JSObject Methods

#### 9.2.1.15.2.2  JSObject.FireEvent Method

```csharp
public void FireEvent(string Name, IJSArguments Arguments);
```

**Description**

This is FireEvent, a member of class JSObject.

**Group**

JSObject Methods

#### 9.2.1.15.2.3  JSObject.ApplyChanges Method

When this method called, all properties getters are internally called looking for changes. Any change to the property value is sent to the client.

```csharp
public void ApplyChanges();
```

**Group**

JSObject Methods
9.2.1.15.2.4  JSObject.ApplyModel Method

Propagates the whole JSObject definition to the javascript client.

C#
public void ApplyModel();

Group
JSObject Methods

9.2.1.15.3  JSObject Properties

The properties of the JSObject class are listed here.

Class
JSObject Class

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Identifier of the object. It must be unique among siblings objects.</td>
</tr>
<tr>
<td>Properties</td>
<td>List containing all properties of this object.</td>
</tr>
<tr>
<td>Methods</td>
<td>List containing all methods of this object.</td>
</tr>
<tr>
<td>Events</td>
<td>List containing all events of this object.</td>
</tr>
<tr>
<td>Objects</td>
<td>List containing all events of this object.</td>
</tr>
</tbody>
</table>

9.2.1.15.3.1  JSObject.Id Property

Identifier of the object. It must be unique among siblings objects.

C#
public string Id;

Group
JSObject Properties

9.2.1.15.3.2  JSObject.Properties Property

List containing all properties of this object.

C#
public IJSProperties Properties;
9.2.1.15.3.3 JSObject.Methods Property

List containing all methods of this object.

C#
public IJSMethods Methods;

9.2.1.15.3.4 JSObject.Events Property

List containing all events of this object.

C#
public IJSEvents Events;

9.2.1.15.3.5 JSObject.Objects Property

List containing all events of this object.

C#
public IJSObjects Objects;

9.2.1.15.4 JSObject Events

The events of the JSObject class are listed here.

Class
JSObject Class

Public Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnExecuteMethod</td>
<td>Fired when a method is executed on the remote object.</td>
</tr>
</tbody>
</table>
9.2.1.15.4.1 JSObject.OnExecuteMethod Event

Fired when a method is executed on the remote object.

**C#**

```csharp
public event EventHandler<JSExecuteMethodEventArgs> OnExecuteMethod;
```

**Group**

JSObject Events

9.2.1.15.4.2 JSObject.OnPropertyChange Event

Fired when a property value has been changed on the remote object.

**C#**

```csharp
public event EventHandler<JSPropertyChangeEventArgs> OnPropertyChange;
```

**Group**

JSObject Events

9.2.1.16 Cybele.Thinfinity.JSPropertySet Type

**C#**

```csharp
public delegate void JSPropertySet(IJSObject Parent, IJSProperty Prop);
```

**File**

Thinfinity.VirtualUI.cs

**Description**

This is type Cybele.Thinfinity.JSPropertySet.

**Namespace**

Cybele.Thinfinity Namespace

9.2.1.17 JSBinding Class

**Class Hierarchy**

```
IJSBinding
    ├── Cybele.Thinfinity.JSBinding
    └── IDisposable
```

© 2015, Cybele Software, Inc.
[ComVisible(true), ClassInterface(ClassInterfaceType.AutoDual), ComDefaultInterface(typeof(IJSBinding))]

```csharp
public class JSBinding : IJSBinding, IDisposable;
```

### File

Thinfinity.VirtualUI.cs

### Description

This is class Cybele.Thinfinity.JSBinding.

### Namespace

Cybele.Thinfinity Namespace

#### 9.2.1.17.1 JSBinding.JSBinding Constructor

```csharp
public JSBinding(JSPropertySet Proc);
```

### Description

This is JSBinding, a member of class JSBinding.

### Class

JSBinding Class

#### 9.2.1.17.2 JSBinding Methods

The methods of the JSBinding class are listed here.

### Class

JSBinding Class

#### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>This is Dispose, a member of class JSBinding.</td>
</tr>
<tr>
<td>Set</td>
<td>This is Set, a member of class JSBinding.</td>
</tr>
</tbody>
</table>

#### 9.2.1.17.2.1 JSBinding.Dispose Method

```csharp
public void Dispose();
```

### Description
This is Dispose, a member of class JSBinding.

- **Group**
  - JSBinding Methods

### 9.2.17.2.2 JSBinding.Set Method

**C#**

```csharp
public void Set(IJSObject Parent, IJSProperty Prop);
```

- **Description**
  
  This is Set, a member of class JSBinding.

- **Group**
  - JSBinding Methods

### 9.2.18 Cybele.Thinfinity.JSMethodCallback Type

**C#**

```csharp
public delegate void JSMethodCallback(IJSObject Parent, IJSMethod Method);
```

- **File**
  
  Thinfinity.VirtualUI.cs

- **Description**
  
  This is type Cybele.Thinfinity.JSMethodCallback.

- **Namespace**
  
  Cybele.Thinfinity Namespace

### 9.2.19 JSCallback Class

- **Class Hierarchy**

  - JSCallback
    - Cybele.Thinfinity.JSMethodCallback
    - IDisposable

  **C#**

  ```csharp
  [ComVisible(true), ClassInterface(ClassInterfaceType.AutoDual), ComDefaultInterface(typeof(IJSCallback))]
  public class JSCallback : IJSCallback, IDisposable;
  ```

- **File**
™ Thinfinity® VirtualUI™

Thinfinity.VirtualUI.cs

- **Description**
  This is class Cybele.Thinfinity.JSCallback.

- **Namespace**
  Cybele.Thinfinity Namespace

### 9.2.1.19.1 JSCallback.JSCallback Constructor

**C#**

```csharp
public JSCallback (JSMethodCallback Proc);
```

- **Description**
  This is JSCallback, a member of class JSCallback.

- **Class**
  JSCallback Class

### 9.2.1.19.2 JSCallback Methods

The methods of the JSCallback class are listed here.

- **Class**
  JSCallback Class

#### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>This is Dispose, a member of class JSCallback.</td>
</tr>
<tr>
<td>Callback</td>
<td>This is Callback, a member of class JSCallback.</td>
</tr>
</tbody>
</table>

### 9.2.1.19.2.1 JSCallback.Dispose Method

**C#**

```csharp
public void Dispose();
```

- **Description**
  This is Dispose, a member of class JSCallback.

- **Group**
JSCallback Methods

9.2.1.19.2.2  JSCallback.Callback Method

```csharp
public void Callback(IJSObject Parent, IJSMethod Method);
```

**Description**

This is Callback, a member of class JSCallback.

**Group**

JSCallback Methods

9.2.2  Cybele.Thinfinity.Settings.VirtualUI Namespace

This is namespace Cybele.Thinfinity.Settings.VirtualUI.

**Classes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Main class. Contains methods and properties to manage all Server configuration.</td>
</tr>
<tr>
<td>ServerUtils</td>
<td>Helper functions.</td>
</tr>
</tbody>
</table>

**Enumerations**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>This is record Cybele.Thinfinity.Settings.VirtualUI.Protocol.</td>
</tr>
<tr>
<td>ProfileKind</td>
<td>This is record Cybele.Thinfinity.Settings.VirtualUI.ProfileKind.</td>
</tr>
<tr>
<td>ScreenResolution</td>
<td>This is record Cybele.Thinfinity.Settings.VirtualUI.ScreenResolution.</td>
</tr>
<tr>
<td>ServerSection</td>
<td>This is record Cybele.Thinfinity.Settings.VirtualUI.ServerSection.</td>
</tr>
</tbody>
</table>

**Group**

.NET Classes

**Interfaces**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRDSSAccounts</td>
<td>Manages the configuration of alternative Remote Desktop Services accounts. VirtualUI makes use of an interactive session. The default setting is to run applications under the console session, but it can be configured to do it under Remote Desktop Services sessions. For the production</td>
</tr>
</tbody>
</table>
9.2.2.1  Cybele.Thinfinity.Settings.VirtualUI.Protocol Enumeration

C#

```csharp
public enum Protocol {
    PROTO_HTTP = 0,
    PROTO_HTTPS = 1
}
```

File
Thinfinity.VirtualUI.Settings.cs

Description
This is record Cybele.Thinfinity.Settings.VirtualUI.Protocol.

Namespace
Cybele.Thinfinity.Settings.VirtualUI Namespace

9.2.2.2  Cybele.Thinfinity.Settings.VirtualUI.ProfileKind Enumeration

C#

```csharp
public enum ProfileKind {
    PROFILE_APP = 0,
    PROFILE_WEBLINK = 1
}
```

File
Thinfinity.VirtualUI.Settings.cs

Description
This is record Cybele.Thinfinity.Settings.VirtualUI.ProfileKind.

Namespace
Cybele.Thinfinity.Settings.VirtualUI Namespace

9.2.2.3  Cybele.Thinfinity.Settings.VirtualUI.ScreenResolution Enumeration

C#

```csharp
public enum ScreenResolution {
}
```

© 2015, Cybele Software, Inc.
SCREENRES_Custom = 0,
SCREENRES_FitToBrowser = 1,
SCREENRES_FitToScreen = 2,
SCREENRES_640x480 = 3,
SCREENRES_800x600 = 4,
SCREENRES_1024x768 = 5,
SCREENRES_1280x720 = 6,
SCREENRES_1280x768 = 7,
SCREENRES_1280x1024 = 8,
SCREENRES_1440x900 = 9,
SCREENRES_1440x1050 = 10,
SCREENRES_1600x1200 = 11,
SCREENRES_1680x1050 = 12,
SCREENRES_1920x1080 = 13,
SCREENRES_1920x1200 = 14

File
Thinfinity.VirtualUI.Settings.cs

Description
This is record Cybele.Thinfinity.Settings.VirtualUI.ScreenResolution.

Namespace
Cybele.Thinfinity.Settings.VirtualUI Namespace

9.2.2.4 Cybele.Thinfinity.Settings.VirtualUI.ServerSection Enumeration

C#
public enum ServerSection {
    SRVSEC_GENERAL = 0,
    SRVSEC_RDS = 1,
    SRVSEC_APPLICATIONS = 2,
    SRVSEC_LICENSES = 3
}

File
Thinfinity.VirtualUI.Settings.cs

Description
This is record Cybele.Thinfinity.Settings.VirtualUI.ServerSection.

Namespace
Cybele.Thinfinity.Settings.VirtualUI Namespace
9.2.2.5 Server Class

Main class. Contains methods and properties to manage all Server configuration.

C#

```csharp
public class Server : VirtualUISLibrary, IServer, IDisposable;
```

File

Thinfinity.VirtualUI.Settings.cs

Namespace

Cybele.Thinfinity.Settings.VirtualUI Namespace

9.2.2.5.1 Server.Server Constructor

C#

```csharp
public Server();
```

Description

This is Server, a member of class Server.

Class

Server Class

9.2.2.5.2 Server Methods

The methods of the Server class are listed here.

Class

Server Class

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>This is Dispose, a member of class Server.</td>
</tr>
<tr>
<td>Load</td>
<td>Loads all the configuration entries and profiles from file. It's automatically called by constructor.</td>
</tr>
</tbody>
</table>
### Save
Saves the entire configuration parameters and profiles.

### HideSection
Hides a configuration section in the VirtualUI Server Manager GUI.

### ShowSection
Makes visible a configuration section in the VirtualUI Server Manager GUI.

#### 9.2.2.5.2.1 Server.Dispose Method

**C#**
```csharp
public void Dispose();
```

**Description**
This is Dispose, a member of class Server.

**Group**
Server Methods

#### 9.2.2.5.2.2 Server.Load Method

Loads all the configuration entries and profiles from file. It's automatically called by constructor.

**C#**
```csharp
public void Load();
```

**Group**
Server Methods

#### 9.2.2.5.2.3 Server.Save Method

Saves the entire configuration parameters and profiles.

**C#**
```csharp
public void Save();
```

**Group**
Server Methods

#### 9.2.2.5.2.4 Server.HideSection Method

Hides a configuration section in the VirtualUI Server Manager GUI.

**C#**
```csharp
public void HideSection(ServerSection section);
```

**Parameters**
Parameters | Description
---|---
ServerSection section | The Server configuration section to hide to user. Use one of the following constants:
- SRVSEC_GENERAL: Hides the General tab, that contains the Binding configuration.
- SRVSEC_RDS: Hides the tab with the Remote Desktop Services account configuration.
- SRVSEC_APPLICATIONS: Hides the list of applications.
- SRVSEC_LICENSES: Hides the tab with License information.

Group
Server Methods

9.2.2.5.2.5 Server.ShowSection Method

Makes visible a configuration section in the VirtualUI Server Manager GUI.

C#

```csharp
public void ShowSection(ServerSection section);
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| ServerSection section | The Server configuration section to show to user. Use one of the following constants:
- SRVSEC_GENERAL: Shows the General tab, that contains the Binding configuration.
- SRVSEC_RDS: Shows the tab with the Remote Desktop Services account configuration.
- SRVSEC_APPLICATIONS: Shows the list of applications.
- SRVSEC_LICENSES: Shows the tab with License information. |

Group
Server Methods

9.2.2.5.3 Server Properties

The properties of the Server class are listed here.

Class
Server Class

Public Properties

| Name | Description |
9.2.2.5.3.1 Server.Binding Property

Returns the Server's Binding configuration.

C#
public IBinding Binding;

See Also
IBinding interface

Group
Server Properties

9.2.2.5.3.2 Server.Certificate Property

Returns the Server's certificate configuration for SSL protocol.

C#
public ICertificate Certificate;

See Also
ICertificate interface

Group
Server Properties

9.2.2.5.3.3 Server.RDSAccounts Property

Returns the list of Remote Desktop Services accounts.

C#
public IRDSAccounts RDSAccounts;

See Also
IRDSAccounts interface
9.2.2.5.3.4 Server.Profiles Property

Returns the profiles list.

C#  
public IProfiles Profiles;

See Also
IProfiles interface

9.2.2.5.3.5 Server.License Property

Returns the current Server's licence.

C#  
public ILicense License;

See Also
ILicense interface

9.2.2.6 ServerUtils Class

Helper functions.

Class Hierarchy

C#  
public class ServerUtils;

File
Thinfinity.VirtualUI.Settings.cs
Namespace

Cybele.Thinfinity.Settings.VirtualUI Namespace

9.2.2.6.1 ServerUtils Methods

The methods of the ServerUtils class are listed here.

Class

ServerUtils Class

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RunAsAdmin</td>
<td>Runs an application in elevated mode. This mode is required to save the Server's configuration in protected files.</td>
</tr>
<tr>
<td>Base64ToIcon</td>
<td>Converts the IProfile.IconData (base64 string) to a PNG image.</td>
</tr>
<tr>
<td>IconToBase64</td>
<td>Converts a PNG image to be stored in IProfile.IconData (as base64 string).</td>
</tr>
</tbody>
</table>

9.2.2.6.1.1 ServerUtils.RunAsAdmin Method

Runs an application in elevated mode. This mode is required to save the Server's configuration in protected files.

C#

```csharp
public static void RunAsAdmin(string fileName, string parameters);
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Full path of application to execute.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Arguments.</td>
</tr>
</tbody>
</table>

Example

In the main program of the application using this classes, you can include: if (args.Length == 0) ServerUtils.RunAsAdmin(Application.ExecutablePath, "/edit"); else { [...] }

Group

ServerUtils Methods

9.2.2.6.1.2 ServerUtils.Base64ToIcon Method

Converts the IProfile.IconData (base64 string) to a PNG image.
C#  
public static Image Base64ToIcon(string data);

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string data</td>
<td>The image encoded in base64.</td>
</tr>
</tbody>
</table>

### Group

ServerUtils Methods

**9.2.2.6.1.3 ServerUtils.iconToBase64 Method**

Converts a PNG image to be stored in IProfile.IconData (as base64 string).

C#  
public static string IconToBase64(Image png);

### Description

param name="png"> The image to be encoded in base64.

### Group

ServerUtils Methods

**9.2.2.7 IRDSAccounts Interface**

Manages the configuration of alternative Remote Desktop Services accounts. VirtualUI makes use of an interactive session. The default setting is to run applications under the console session, but it can be configured to do it under Remote Desktop Services sessions. For the production environment, it is recommended to set VirtualUI to run applications under its own Remote Desktop Services session. This will ensure that the service is available at all times. Alternatively, you can choose to have VirtualUI run the applications under the console session by configuring the Auto Logon feature on your Windows operating system.

### Class Hierarchy

C#  
[Guid("60666BC2-7E17-4842-9716-CFA3DFD5583"), InterfaceType(ComInterfaceType.InterfaceIsIDispatch)]  
public interface IRDSAccounts;

### File

Thinfinity.VirtualUI.Settings.cs

### Namespace

Cybele.Thinfinity.Settings.VirtualUI Namespace
9.2.2.7.1 IRDSAccounts Methods

The methods of the IRDSAccounts class are listed here.

- **Interface**
  
  IRDSAccounts Interface

- **Public Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Creates a new RDS account and adds it to the list. If CreateAccount is true, the account will be created in your system.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes an RDS account from the list. If DeleteAccount is true, the account will be delete in your system.</td>
</tr>
</tbody>
</table>

9.2.2.7.1.1 IRDSAccounts.Add Method

Creates a new RDS account and adds it to the list. If CreateAccount is true, the account will be created in your system.

**C#**

```csharp
IRDS Add(string UserName, string Password, bool CreateAccount);
```

- **Returns**
  
  The newly created RDS account.

- **See Also**
  
  IRDS interface

- **Group**
  
  IRDSAccounts Methods

9.2.2.7.1.2 IRDSAccounts.Delete Method

Deletes an RDS account from the list. If DeleteAccount is true, the account will be delete in your system.

**C#**

```csharp
bool Delete(string UserName, bool DeleteAccount);
```

- **Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rds</td>
<td>The account to be deleted.</td>
</tr>
</tbody>
</table>

- **Group**
IRDSAccounts Methods

9.2.2.7.2 IRDSAccounts Indexers

The indexers of the IRDSAccounts class are listed here.

Interface
IRDSAccounts Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>Returns an RDS account from the list by its index.</td>
</tr>
</tbody>
</table>

9.2.2.7.2.1 IRDSAccounts.this Indexer

Returns an RDS account from the list by its index.

C#

IRDS this[int index];

See Also
IRDS interface

Group
IRDSAccounts Indexers

9.2.2.7.3 IRDSAccounts Properties

The properties of the IRDSAccounts class are listed here.

Interface
IRDSAccounts Interface

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Returns the accounts count.</td>
</tr>
</tbody>
</table>

9.2.2.7.3.1 IRDSAccounts.Count Property

Returns the accounts count.

C#
```
int Count;
```

## Group

**IRDSAccounts Properties**

### 9.3 Delphi Classes

#### Namespaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualUI_SDK</td>
<td>This is namespace VirtualUI_SDK.</td>
</tr>
<tr>
<td>VirtualUI_Settings</td>
<td></td>
</tr>
</tbody>
</table>

#### VirtualUI_SDK Namespace

This is namespace VirtualUI_SDK.

#### Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBrowserInfo</td>
<td>Contains information regarding the end-user’s screen, web browser, the window containing VirtualUI Viewer and VirtualUI Viewer itself. The VirtualUI Viewer runs inside an HTML DIV element contained in a frame of browser window on the end-user's application page.</td>
</tr>
<tr>
<td>TClientSettings</td>
<td>Allows to set some client settings.</td>
</tr>
<tr>
<td>TDevServer</td>
<td>Contains properties to manage the VirtualUI Development Server as well as the access from the developer's web browser.</td>
</tr>
<tr>
<td>TEventSink</td>
<td>This is class VirtualUI_SDK.TEventSink.</td>
</tr>
<tr>
<td>TJSBinding</td>
<td>This is class VirtualUI_SDK.TJSBinding.</td>
</tr>
<tr>
<td>TJSCallback</td>
<td>This is class VirtualUI_SDK.TJSCallback.</td>
</tr>
<tr>
<td>TJSObject</td>
<td>Represents a custom remotable object.</td>
</tr>
<tr>
<td>TJSObjectEventSink</td>
<td>This is class VirtualUI_SDK.TJSObjectEventSink.</td>
</tr>
<tr>
<td>TVirtualUI</td>
<td>Main class. Has methods, properties and events to allow the activation and control the behavior of VirtualUI.</td>
</tr>
</tbody>
</table>

#### Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDllDir</td>
<td>Returns the path where Thinfinity.VirtualUI.dll is located.</td>
</tr>
<tr>
<td>VirtualUI</td>
<td>Returns a global VirtualUI object.</td>
</tr>
</tbody>
</table>
### Group

**Delphi Classes**

#### 9.3.1.1 TBrowserInfo Class

Contains information regarding the end-user's screen, web browser, the window containing VirtualUI Viewer and VirtualUI Viewer itself. The VirtualUI Viewer runs inside an HTML DIV element contained in a frame of browser window on the end-user's application page.

### Class Hierarchy

```
TInterfacedObject

IBrowserInfo

VirtualUI_SDK:TBrowserInfo
```

Pascal

```
TBrowserInfo = class(TInterfacedObject, IBrowserInfo);
```

### File

VirtualUI_SDK

### Namespace

**VirtualUI_SDK Namespace**

#### 9.3.1.1.1 TBrowserInfo.Create Constructor

```
Pascal
constructor Create(AVirtualUI: IVirtualUI);
```

### Description

This is Create, a member of class TBrowserInfo.

### Class

**TBrowserInfo Class**

#### 9.3.1.1.2 TBrowserInfo.Destroy Destructor

```
Pascal
destructor Destroy; override;
```

### Description

This is Destroy, a member of class TBrowserInfo.

### Class
### TBrowserInfo Class

9.3.1.1.3  TBrowserInfo Properties

The properties of the TBrowserInfo class are listed here.

#### Class

TBrowserInfo Class

#### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewWidth</td>
<td>Returns the width of the VirtualUI Viewer.</td>
</tr>
<tr>
<td>ViewHeight</td>
<td>Returns the height of the VirtualUI Viewer.</td>
</tr>
<tr>
<td>BrowserWidth</td>
<td>Returns the width of the HTML element containing the VirtualUI Viewer.</td>
</tr>
<tr>
<td>BrowserHeight</td>
<td>Returns the height of the HTML element containing the VirtualUI Viewer.</td>
</tr>
<tr>
<td>ScreenWidth</td>
<td>Returns the width of the end-user's monitor screen.</td>
</tr>
<tr>
<td>ScreenHeight</td>
<td>Returns the height of the end-user's monitor screen.</td>
</tr>
<tr>
<td>Username</td>
<td>Returns the logged-on Username.</td>
</tr>
<tr>
<td>IPAddress</td>
<td>Returns the client's IP address.</td>
</tr>
<tr>
<td>UserAgent</td>
<td>Returns the browser's User Agent string.</td>
</tr>
<tr>
<td>ScreenResolution</td>
<td>Returns the application screen resolution defined in the application profile.</td>
</tr>
<tr>
<td>Orientation</td>
<td>Returns the browser's orientation.</td>
</tr>
<tr>
<td>UniqueBrowserId</td>
<td>UniqueBrowserId identifies an instance of a Web Browser. Each time an end-user opens the application from a different browser window, this ID will have a different value.</td>
</tr>
<tr>
<td>Location</td>
<td>Returns the URL of the current application.</td>
</tr>
</tbody>
</table>

9.3.1.1.3.1  TBrowserInfo.ViewWidth Property

Returns the width of the VirtualUI Viewer.

**Pascal**

```
property ViewWidth: Integer;
```
9.3.1.1.3.2 TBrowserInfo.ViewHeight Property

Returns the height of the VirtualUI Viewer.

Pascal
property ViewHeight: Integer;

Group
TBrowserInfo Properties

9.3.1.1.3.3 TBrowserInfo.BrowserWidth Property

Returns the width of the HTML element containing the VirtualUI Viewer.

Pascal
property BrowserWidth: Integer;

Group
TBrowserInfo Properties

9.3.1.1.3.4 TBrowserInfo.BrowserHeight Property

Returns the height of the HTML element containing the VirtualUI Viewer.

Pascal
property BrowserHeight: Integer;

Group
TBrowserInfo Properties

9.3.1.1.3.5 TBrowserInfo.ScreenWidth Property

Returns the width of the end-user's monitor screen.

Pascal
property ScreenWidth: Integer;

Group
TBrowserInfo Properties

9.3.1.1.3.6 TBrowserInfo.ScreenHeight Property

Returns the height of the end-user's monitor screen.

Pascal
property ScreenHeight: Integer;
9.3.1.1.3.7 TBrowserInfo.Username Property

Returns the logged-on Username.

Pascal
property Username: WideString;

9.3.1.1.3.8 TBrowserInfo.IPAddress Property

Returns the client's IP address.

Pascal
property IPAddress: WideString;

9.3.1.1.3.9 TBrowserInfo.UserAgent Property

Returns the browser’s User Agent string.

Pascal
property UserAgent: WideString;

9.3.1.1.3.10 TBrowserInfo.ScreenResolution Property

Returns the application screen resolution defined in the application profile.

Pascal
property ScreenResolution: Integer;
9.3.1.1.3.11 TBrowserInfo.Orientation Property

Returns the browser’s orientation.

Pascal

property Orientation: Orientation;

Group

TBrowserInfo Properties

9.3.1.1.3.12 TBrowserInfo.UniqueBrowserId Property

UniqueBrowserId identifies an instance of a Web Browser. Each time an end-user opens the application from a different browser window, this ID will have a different value.

Pascal

property UniqueBrowserId: WideString;

Group

TBrowserInfo Properties

9.3.1.1.3.13 TBrowserInfo.Location Property

Returns the URL of the current application.

Pascal

property Location: WideString;

Group

TBrowserInfo Properties

9.3.1.2 TClientSettings Class

Allows to set some client settings.

Class Hierarchy

Pascal

TClientSettings = class(TInterfacedObject, IClientSettings);

File

VirtualUI_SDK
9.3.1.2.1  TClientSettings.Create Constructor

**Pascal**

```
constructor Create(AVirtualUI: IVirtualUI);
```

**Description**

This is Create, a member of class TClientSettings.

**Class**

TClientSettings Class

9.3.1.2.2  TClientSettings.Destroy Destructor

**Pascal**

```
destructor Destroy; override;
```

**Description**

This is Destroy, a member of class TClientSettings.

**Class**

TClientSettings Class

9.3.1.2.3  TClientSettings Properties

The properties of the TClientSettings class are listed here.

**Class**

TClientSettings Class

**Public Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MouseMoveGestureStyle</td>
<td>Valid for touch devices. Specifies whether the mouse pointer is shown and acts on the exact spot of the finger touch (absolute) or its position is managed relatively to the movement of the finger touch (relative).</td>
</tr>
<tr>
<td>MouseMoveGestureAction</td>
<td>Specifies whether the &quot;mouse move&quot; simulation on a touch device is interpreted as a mouse move or as a mouse wheel.</td>
</tr>
<tr>
<td>CursorVisible</td>
<td>Hides/shows the mouse pointer.</td>
</tr>
</tbody>
</table>
9.3.1.2.3.1 TClientSettings.MouseMoveGestureStyle Property

Valid for touch devices. Specifies whether the mouse pointer is shown and acts on the exact spot of the finger touch (absolute) or its position is managed relatively to the movement of the finger touch (relative).

Pascal

property MouseMoveGestureStyle: MouseMoveGestureStyle;

Group

TClientSettings Properties

9.3.1.2.3.2 TClientSettings.MouseMoveGestureAction Property

Specifies whether the "mouse move" simulation on a touch device is interpreted as a mouse move or as a mouse wheel.

Pascal

property MouseMoveGestureAction: MouseMoveGestureAction;

Group

TClientSettings Properties

9.3.1.2.3.3 TClientSettings.CursorVisible Property

Hides/shows the mouse pointer.

Pascal

property CursorVisible: WordBool;

Group

TClientSettings Properties

9.3.1.3 TDevServer Class

Contains properties to manage the VirtualUI Development Server as well as the access from the developer's web browser.

Class Hierarchy

Pascal

TDevServer = class(TInterfacedObject, IDevServer);
VirtualUI_SDK

 Namespace
VirtualUI_SDK Namespace

9.3.1.3.1 TDevServer.Create Constructor

 Pascal
constructor Create(AVirtualUI: IVirtualUI);

 Description
This is Create, a member of class TDevServer.

 Class
TDevServer Class

9.3.1.3.2 TDevServer.Destroy Destructor

 Pascal
destructor Destroy; override;

 Description
This is Destroy, a member of class TDevServer.

 Class
TDevServer Class

9.3.1.3.3 TDevServer Properties

The properties of the TDevServer class are listed here.

 Class
TDevServer Class

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Enables/disables the Development Server.</td>
</tr>
<tr>
<td>Port</td>
<td>Gets/sets the Development Server's TCP/IP listening port.</td>
</tr>
<tr>
<td>StartBrowser</td>
<td>Instructs VirtualUI whether start or not the local web browser upon VirtualUI activation.</td>
</tr>
</tbody>
</table>
9.3.1.3.3.1 TDevServer.Enabled Property

Enables/disables the Development Server.

Pascal

property Enabled: WordBool;

Group

TDevServer Properties

9.3.1.3.3.2 TDevServer.Port Property

Gets/sets the Development Server's TCP/IP listening port.

Pascal

property Port: Integer;

Group

TDevServer Properties

9.3.1.3.3.3 TDevServer.StartBrowser Property

Instructs VirtualUI whether start or not the local web browser upon VirtualUI activation.

Pascal

property StartBrowser: WordBool;

Group

TDevServer Properties

9.3.1.4 TEventSink Class

Class Hierarchy

Pascal

TEventSink = class(TInterfacedObject, IUnknown, IDispatch);

File

VirtualUI_SDK

Description
This is class VirtualUI_SDK.TEventSink.

### Namespace

VirtualUI_SDK Namespace

#### 9.3.1.4.1 TEventSink.Create Constructor

**Pascal**

```pascal
class Create(Controller: TObject);
```

### Description

This is Create, a member of class TEventSink.

### Class

TEventSink Class

#### 9.3.1.4.2 TEventSink.Destroy Destructor

**Pascal**

```pascal
deructor Destroy; override;
```

### Description

This is Destroy, a member of class TEventSink.

### Class

TEventSink Class

#### 9.3.1.5 TJSBinding Class

### Class Hierarchy

![Class Hierarchy Diagram]

**Pascal**

```pascal
TJSBinding = class(TInterfacedObject, IJSBinding);
```

### File

VirtualUI_SDK

### Description

This is class VirtualUI_SDK.TJSBinding.
Namespace

VirtualUI_SDK Namespace

9.3.1.5.1 Create Constructor

This is the overview for the Create constructor overload.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TJSBinding.Create(TJSPropertyNNProc)</td>
<td>Creates an IJSBinding wrapper passing an anonymous procedure as a callback.</td>
</tr>
<tr>
<td>TJSBinding.Create(TJSPropertyProc)</td>
<td>Creates an IJSBinding wrapper passing an object procedure as a callback.</td>
</tr>
</tbody>
</table>

Class

TJSBinding Class

9.3.1.5.1.1 TJSBinding.Create Constructor (TJSPropertyNNProc)

Creates an IJSBinding wrapper passing an anonymous procedure as a callback.

Pascal

```
constructor Create(ANNProc: TJSPropertyNNProc); overload;
```

Group

Create Constructor

9.3.1.5.1.2 TJSBinding.Create Constructor (TJSPropertyProc)

Creates an IJSBinding wrapper passing an object procedure as a callback.

Pascal

```
constructor Create(AProc: TJSPropertyProc); overload;
```

Group

Create Constructor

9.3.1.6 TJSCallback Class

Class Hierarchy

```
TJSCallback

```

Pascal

© 2015, Cybele Software, Inc.
File
VirtualUI_SDK

Description
This is class VirtualUI_SDK.TJSCallback.

Namespace
VirtualUI_SDK Namespace

9.3.1.6.1 Create Constructor

This is the overview for the Create constructor overload.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TJSCallback.Create(TJSMethodNNProc)</td>
<td>Creates an IJSCallback wrapper passing an anonymous procedure as a callback.</td>
</tr>
<tr>
<td>TJSCallback.Create(TJSMethodProc)</td>
<td>Creates an IJSCallback wrapper passing an object procedure as a callback.</td>
</tr>
</tbody>
</table>

Class

TJSCallback Class

9.3.1.6.1.1 TJSCallback.Create Constructor (TJSMethodNNProc)

Creates an IJSCallback wrapper passing an anonymous procedure as a callback.

Pascal
class constructor Create(ANNProc: TJSMethodNNProc); overload;

Group
Create Constructor

9.3.1.6.1.2 TJSCallback.Create Constructor (TJSMethodProc)

Creates an IJSCallback wrapper passing an object procedure as a callback.

Pascal
class constructor Create(AProc: TJSMethodProc); overload;

Group
Create Constructor

9.3.1.7 TJSObject Class

Represents a custom remotable object.

Class Hierarchy

![Class Hierarchy Diagram]

Pascal

\[ \text{TJSObject} = \text{class}(\text{TInterfacedObject}, \text{IJSOObject}); \]

File

VirtualUI_SDK

Remarks

TJSObject allows you to define an object model that is mirrored on the client side, and allows for an easy, powerful and straight-forward way to connect the web browser client application and the remoted Windows application.

TJSObject can contain properties (IJSProperties), methods (IJSMethods), events (IJSEvents) and children objects. Changes to properties values are propagated in from server to client and viceversa, keeping the data synchronized.

TJSObject is defined as a model seen from the client perspective. A method (IJSMethod) is called on the client side and executed on the server side. An event (IJSEvent) is called on the server side and raised on the client side.

Namespace

VirtualUI_SDK Namespace

9.3.1.7.1 TJSObject.Create Constructor

Pascal

\[ \text{constructor Create(const Id: WideString);} \]

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Id: WideString</td>
<td><em>nt</em></td>
</tr>
</tbody>
</table>

Description

Class

TJSObject Class
9.3.1.7.2  TJSObject.Destroy Destructor

Pascal
destructor Destroy; override;

Description
This is Destroy, a member of class TJSObject.

Class
TJSObject Class

9.3.1.7.3  TJSObject Methods

The methods of the TJSObject class are listed here.

Class
TJSObject Class

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FireEvent</td>
<td>Fires the event specified in Name on the client-size javascript API.</td>
</tr>
<tr>
<td>ApplyChanges</td>
<td>When this method called, all properties getters are internally called looking for changes. Any change to the property value is sent to the client.</td>
</tr>
<tr>
<td>ApplyModel</td>
<td>Propagates the whole JSObject definition to the javascript client.</td>
</tr>
</tbody>
</table>

9.3.1.7.3.1  TJSObject.FireEvent Method

Fires the event specified in Name on the client-size javascript API.

Pascal
procedure FireEvent(const Name: WideString; const Arguments: IJSArguments); safecall;

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Name: WideString</td>
<td>Event name</td>
</tr>
<tr>
<td>const Arguments: IJSArguments</td>
<td>List of arguments</td>
</tr>
</tbody>
</table>

Group
TJSObject Methods
9.3.1.7.3.2 TJSObject.ApplyChanges Method

When this method called, all properties getters are internally called looking for changes. Any change to the property value is sent to the client.

Pascal

procedure ApplyChanges; safecall;

Group

TJSObject Methods

9.3.1.7.3.3 TJSObject.ApplyModel Method

Propagates the whole JSObject definition to the javascript client.

Pascal

procedure ApplyModel; safecall;

Group

TJSObject Methods

9.3.1.7.4 TJSObject Properties

The properties of the TJSObject class are listed here.

Class

TJSObject Class

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Identifier of the object. It must be unique among siblings objects.</td>
</tr>
<tr>
<td>Properties</td>
<td>List containing all properties of this object.</td>
</tr>
<tr>
<td>Methods</td>
<td>List containing all methods of this object.</td>
</tr>
<tr>
<td>Events</td>
<td>List containing all events of this object.</td>
</tr>
<tr>
<td>Objects</td>
<td>List containing all events of this object.</td>
</tr>
</tbody>
</table>

9.3.1.7.4.1 TJSObject.Id Property

Identifier of the object. It must be unique among siblings objects.

Pascal

property Id: WideString;
9.3.1.7.4.2  TJSObject.Properties Property

List containing all properties of this object.

Pascal

```pascal
property Properties: IJSProperties;
```
9.3.1.7.5 TJSObject Events

The events of the TJSObject class are listed here.

Class
TJSObject Class

Public Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnExecuteMethod</td>
<td>Fired when a method is executed on the remote object.</td>
</tr>
<tr>
<td>OnPropertyChange</td>
<td>Fired when a property value has been changed on the remote object.</td>
</tr>
</tbody>
</table>

9.3.1.7.5.1 TJSObject.OnExecuteMethod Event

Fired when a method is executed on the remote object.

Pascal

```pascal
property OnExecuteMethod: TExecuteMethodEvent;
```

Group
TJSObject Events

9.3.1.7.5.2 TJSObject.OnPropertyChange Event

Fired when a property value has been changed on the remote object.

Pascal

```pascal
property OnPropertyChange: TPropertyChangeEvent;
```

Group
TJSObject Events

9.3.1.8 TJSObjectEventSink Class

Class Hierarchy

Pascal

```
TJSObjectEventSink = class(TEventSink);
```
File
VirtualUI_SDK

Description
This is class VirtualUI_SDK.TJSObjectEventSink.

Namespace
VirtualUI_SDK Namespace

9.3.1.9 TVirtualUI Class

Main class. Has methods, properties and events to allow the activation and control the behavior of VirtualUI.

Class Hierarchy

```
  TInterfacedObject
    TVirtualUI
```

Pascal

```
TVirtualUI = class(TInterfacedObject, IVirtualUI);
```

File
VirtualUI_SDK

Namespace
VirtualUI_SDK Namespace

9.3.1.9.1 TVirtualUI.Create Constructor

```
Pascal
constructor Create;
```

Description
This is Create, a member of class TVirtualUI.

Class
TVirtualUI Class
9.3.1.9.2 TVirtualUI.Destroy Destructor

**Pascal**
destructor Destroy; override;

**Description**

This is Destroy, a member of class TVirtualUI.

**Class**

TVirtualUI Class

9.3.1.9.3 TVirtualUI Methods

The methods of the TVirtualUI class are listed here.

**Class**

TVirtualUI Class

**Public Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DownloadFile</td>
<td>This is the overview for the DownloadFile method overload.</td>
</tr>
<tr>
<td>Start</td>
<td>This is the overview for the Start method overload.</td>
</tr>
<tr>
<td>UploadFile</td>
<td>This is the overview for the UploadFile method overload.</td>
</tr>
<tr>
<td>Stop</td>
<td>Deactivates VirtualUI, closing the connection with the end-user's web browser.</td>
</tr>
<tr>
<td>PrintPdf</td>
<td>Sends the specified PDF file to be shown on the end-user's web browser.</td>
</tr>
<tr>
<td>TakeScreenshot</td>
<td>Takes a screenshot of a Window.</td>
</tr>
<tr>
<td>OpenLinkDlg</td>
<td>Displays a popup with a button to open a web link.</td>
</tr>
<tr>
<td>SendMessage</td>
<td>Sends a data string to the web browser.</td>
</tr>
<tr>
<td>AllowExecute</td>
<td>Allows the execution of the passed application.</td>
</tr>
<tr>
<td>SetImageQualityByWnd</td>
<td>Allows to the the image quality for the specified window.</td>
</tr>
<tr>
<td>ShowVirtualKeyboard</td>
<td>In mobile, shows the keyboard.</td>
</tr>
</tbody>
</table>

9.3.1.9.3.1 DownloadFile Method

This is the overview for the DownloadFile method overload.
## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TVirtualUI.DownloadFile (WideString)</td>
<td>Sends the specified file to the end-user's web browser for saving it in the remote machine.</td>
</tr>
<tr>
<td>TVirtualUI.DownloadFile (WideString, WideString)</td>
<td>Sends the specified file to the end-user's web browser for saving it in the remote machine.</td>
</tr>
<tr>
<td>TVirtualUI.DownloadFile (WideString, WideString, WideString)</td>
<td>Sends the specified file to the end-user's web browser for saving it in the remote machine.</td>
</tr>
</tbody>
</table>

## Group

**TVirtualUI Methods**

Sends the specified file to the end-user's web browser for saving it in the remote machine.

**Pascal**

```pascal
procedure DownloadFile(const LocalFilename: WideString; const RemoteFilename: WideString; const MimeType: WideString);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const LocalFilename: WideString</td>
<td>Name of the local file to be sent.</td>
</tr>
<tr>
<td>const RemoteFilename: WideString</td>
<td>Name of the file in the remote machine.</td>
</tr>
<tr>
<td>const MimeType: WideString</td>
<td>content-type of the file. If specified, the content will be handled by browser. Leave blank to force download.</td>
</tr>
</tbody>
</table>

## Group

**DownloadFile Method**

Sends the specified file to the end-user's web browser for saving it in the remote machine.

**Pascal**

```pascal
procedure DownloadFile(const LocalFilename: WideString; const RemoteFilename: WideString); safecall
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const LocalFilename: WideString</td>
<td>Name of the local file to be sent.</td>
</tr>
<tr>
<td>const RemoteFilename: WideString</td>
<td>Name of the file in the remote machine.</td>
</tr>
</tbody>
</table>
Sends the specified file to the end-user’s web browser for saving it in the remote machine.

**Pascal**

```pascal
procedure DownloadFile(const LocalFilename: WideString); overload;
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const LocalFilename: WideString</td>
<td>Name of both the local and remote file.</td>
</tr>
</tbody>
</table>

**Group**

DownloadFile Method

#### 9.3.1.9.3.2 Start Method

This is the overview for the Start method overload.

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TVirtualUI.Start()</td>
<td>Starts the VirtualUI’s activation process. Returns true if VirtualUI was fully activated or false if the timeout expired. The timeout is 60 seconds.</td>
</tr>
<tr>
<td>TVirtualUI.Start(Integer)</td>
<td>Starts the VirtualUI’s activation process. Returns true if VirtualUI was fully activated or false if the passed timeout expired.</td>
</tr>
</tbody>
</table>

**Group**

TVirtualUI Methods

Starts the VirtualUI’s activation process. Returns true if VirtualUI was fully activated or false if the passed timeout expired.

**Pascal**

```pascal
function Start(Timeout: Integer): WordBool; safecall; overload;
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout: Integer</td>
<td>Maximum time, in seconds, until the activation process is canceled. Defaults to 60 seconds.</td>
</tr>
</tbody>
</table>

**Remarks**

To fully activate VirtualUI, the connection with the end-user’s web browser must established within the time specified by Timeout parameter.

**Group**
**Start Method**

Starts the VirtualUI's activation process. Returns true if VirtualUI was fully activated or false if the timeout expired. The timeout is 60 seconds.

**Pascal**

```pascal
function Start: WordBool; overload;
```

**UploadFile Method**

This is the overview for the UploadFile method overload.

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TVirtualUI.UploadFile ()</td>
<td>Selects a file from client machine, and it's uploaded to VirtualUI public path.</td>
</tr>
<tr>
<td>TVirtualUI.UploadFile (WideString)</td>
<td>Selects a file from client machine, and it's uploaded to ServerDirectory</td>
</tr>
</tbody>
</table>

**Group**

**TVirtualUI Methods**

Selects a file from client machine, and it's uploaded to ServerDirectory

**Pascal**

```pascal
procedure UploadFile(const ServerDirectory: WideString); safecall; overload;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>

**Group**

**UploadFile Method**

Selects a file from client machine, and it's uploaded to VirtualUI public path.

**Pascal**

```pascal
procedure UploadFile; overload;
```
UploadFile Method

9.3.1.9.3.4 TVirtualUI.Stop Method

Deactivates VirtualUI, closing the connection with the end-user’s web browser.

Pascal

procedure Stop; safecall;

Group

TVirtualUI Methods

9.3.1.9.3.5 TVirtualUI.PrintPdf Method

Sends the specified PDF file to be shown on the end-user’s web browser.

Pascal

procedure PrintPdf(const AFileName: WideString); safecall;

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const AFileName: WideString</td>
<td>Name of the PDF file.</td>
</tr>
</tbody>
</table>

Remarks

PrintPDF is similar to DownloadFile, except that it downloads the file with a content-type: application/pdf.

Group

TVirtualUI Methods

9.3.1.9.3.6 TVirtualUI.TakeScreenshot Method

Takes a screenshot of a Window.

Pascal

function TakeScreenshot(Wnd: Integer; const FileName: WideString): WordBool; safecall;

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wnd: Integer</td>
<td>The Window to capture.</td>
</tr>
<tr>
<td>const FileName: WideString</td>
<td>Full path of file to save screenshot. Extensions allowed: jpg, bmp, png.</td>
</tr>
</tbody>
</table>

Group
TVirtualUI Methods

9.3.1.9.3.7 TVirtualUI.OpenLinkDlg Method

Displays a popup with a button to open a web link.

**Pascal**

```pascal
procedure OpenLinkDlg(const url: WideString; const caption: WideString); safecall;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const url: WideString</td>
<td>Link to open.</td>
</tr>
<tr>
<td>const caption: WideString</td>
<td>Text to display in popup.</td>
</tr>
</tbody>
</table>

**Remarks**

This method is used to send custom data to the browser for custom purposes.

9.3.1.9.3.8 TVirtualUI.SendMessage Method

Sends a data string to the web browser.

**Pascal**

```pascal
procedure SendMessage(const Data: WideString); safecall;
```

**Remarks**

This method is used to send custom data to the browser for custom purposes.

9.3.1.9.3.9 TVirtualUI.AllowExecute Method

Allows the execution of the passed application.

**Pascal**

```pascal
procedure AllowExecute(const Filename: WideString); safecall;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Filename: WideString</td>
<td>regular expression specifying the filename(s) of the applications allowed to run.</td>
</tr>
</tbody>
</table>

**Remarks**
Under **VirtualUI** environment, only applications precompiled with **VirtualUI** SDK should be allowed to run. Applications not under **VirtualUI** control, cannot be controlled.

**Group**

**TVirtualUI Methods**

### 9.3.1.9.3.10 TVirtualUI.SetImageQualityByWnd Method

Allows to the the image quality for the specified window.

**Pascal**

```pascal
procedure SetImageQualityByWnd(Wnd: Integer; const Class_: WideString; Quality: Integer); safecall;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wnd: Integer</td>
<td>Window handle.</td>
</tr>
<tr>
<td>Quality: Integer</td>
<td>Quality from 0 to 100.</td>
</tr>
<tr>
<td>Class</td>
<td>Window classname.</td>
</tr>
</tbody>
</table>

**Group**

**TVirtualUI Methods**

### 9.3.1.9.3.11 TVirtualUI.ShowVirtualKeyboard Method

In mobile, shows the keyboard.

**Pascal**

```pascal
procedure ShowVirtualKeyboard; safecall;
```

**Group**

**TVirtualUI Methods**

### 9.3.1.9.4 TVirtualUI Properties

The properties of the TVirtualUI class are listed here.

**Class**

**TVirtualUI Class**

**Public Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Returns the <strong>VirtualUI</strong>’s state.</td>
</tr>
</tbody>
</table>
9.3.1.9.4.1 TVirtualUI.Active Property

Returns the VirtualUI’s state.

**Pascal**

\[
\text{property Active: WordBool;}
\]

**Remarks**
When in development mode, applications executed under the IDE, connect to the Development Server, allowing the access to the application from the browser while in debugging.

9.3.1.9.4.2 TVirtualUI.Enabled Property

Enables/disables VirtualUI for the container application.

**Pascal**

\[
\text{property Enabled: WordBool;}
\]

9.3.1.9.4.3 TVirtualUI.DevMode Property

Gets/sets the development mode.

**Pascal**

\[
\text{property DevMode: WordBool;}
\]
9.3.1.9.4.4 TVirtualUI.StdDialogs Property

Enables/disables the use of standard dialogs.

**Pascal**

```
property StdDialogs: WordBool;
```

**Remarks**

When set to false, the standard save, open and print dialogs are replaced by native browser ones, enabling you to extend the operations to the remote computer.

**Group**

[TVirtualUI Properties](#)

9.3.1.9.4.5 TVirtualUI.BrowserInfo Property

Contains information regarding the end-user's environment.

**Pascal**

```
property BrowserInfo: IBrowserInfo;
```

**Group**

[TVirtualUI Properties](#)

9.3.1.9.4.6 TVirtualUI.DevServer Property

Allows for managing the Development Server.

**Pascal**

```
property DevServer: IDevServer;
```

**Group**

[TVirtualUI Properties](#)

9.3.1.9.4.7 TVirtualUI.ClientSettings Property

Controls some working parameters on the client side.

**Pascal**

```
property ClientSettings: IClientSettings;
```

**Group**

[TVirtualUI Properties](#)
9.3.1.9.5 TVirtualUI Events

The events of the TVirtualUI class are listed here.

Class

TVirtualUI Class

Public Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnGetUploadDir</td>
<td>Fires during an upload request, allowing you to change the save folder.</td>
</tr>
<tr>
<td>OnBrowserResize</td>
<td>Fires when the VirtualUI Viewer's container window resizes. Normally, when the browser resizes.</td>
</tr>
<tr>
<td>OnReceiveMessage</td>
<td>Fires when a custom data string is sent from the web browser page.</td>
</tr>
<tr>
<td>OnClose</td>
<td>Fires when the browser window is about to close.</td>
</tr>
<tr>
<td>OnDownloadEnd</td>
<td>Fires when the file has been sent.</td>
</tr>
</tbody>
</table>

9.3.1.9.5.1 TVirtualUI.OnGetUploadDir Event

Fires during an upload request, allowing you to change the save folder.

Pascal

property OnGetUploadDir: TGetUploadDirEvent;

Group

TVirtualUI Events

9.3.1.9.5.2 TVirtualUI.OnBrowserResize Event

Fires when the VirtualUI Viewer's container window resizes. Normally, when the browser resizes.

Pascal

property OnBrowserResize: TBrowserResizeEvent;

Remarks

Allows you to take action when the VirtualUI Viewer's container window resizes. Set Handled to true to disable the default processing, which resizing all maximized windows.

Group

TVirtualUI Events
9.3.1.9.5.3 TVirtualUI.OnReceiveMessage Event

Fires when a custom data string is sent from the web browser page.

**Pascal**

property OnReceiveMessage: TReceiveMessageEvent;

--- Group

TVirtualUI Events

9.3.1.9.5.4 TVirtualUI.OnClose Event

Fires when the browser window is about to close.

**Pascal**

property OnClose: TCloseEvent;

--- Group

TVirtualUI Events

9.3.1.9.5.5 TVirtualUI.OnDownloadEnd Event

Fires when the file has been sent.

**Pascal**

property OnDownloadEnd: TDownloadEndEvent;

--- Group

TVirtualUI Events

9.3.1.10 VirtualUI_SDK.GetDllDir Function

Returns the path where Thinfinity.VirtualUI.dll is located.

**Pascal**

function GetDllDir: string;

--- File

VirtualUI_SDK

--- Namespace

VirtualUI_SDK Namespace
9.3.1.11 VirtualUI_SDK.VirtualUI Function

Returns a global VirtualUI object.

**Pascal**

```pascal
function VirtualUI: TVirtualUI;
```

**File**

VirtualUI_SDK

**Namespace**

VirtualUI_SDK Namespace

9.3.2 VirtualUI_Settings Unit

**Classes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TServer</td>
<td>Main class. Contains methods and properties to manage all Server configuration.</td>
</tr>
</tbody>
</table>

**Functions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base64ToIcon</td>
<td>Converts the IProfile IconData (base64 string) to a PNG image.</td>
</tr>
<tr>
<td>IconToBase64</td>
<td>Converts a PNG image to be stored in IProfile IconData (as base64 string).</td>
</tr>
</tbody>
</table>

**Group**

Delphi Classes

9.3.2.1 TServer Class

Main class. Contains methods and properties to manage all Server configuration.

**Class Hierarchy**

```
TInterfacedObject
  +-------------
  |            |
  | VirtualUI   |
  +-------------
  | TServer     |
  +-------------

Pascal

TServer = class(TInterfacedObject, IServer);
```

**File**

© 2015, Cybele Software, Inc.
VirtualUI_Settings

■ Namespace

VirtualUI_Settings Unit

9.3.2.1.1 TServer.Create Constructor

Pascal
constructor Create;

■ Description

This is Create, a member of class TServer.

■ Class

TServer Class

9.3.2.1.2 TServer.Destroy Destructor

Pascal
destructor Destroy; override;

■ Description

This is Destroy, a member of class TServer.

■ Class

TServer Class

9.3.2.1.3 TServer Methods

The methods of the TServer class are listed here.

■ Class

TServer Class

■ Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>Loads all the configuration entries and profiles from file. It is automatically called by constructor.</td>
</tr>
<tr>
<td>Save</td>
<td>Saves the entire configuration parameters and profiles.</td>
</tr>
<tr>
<td>HideSection</td>
<td>Hides a configuration section in the VirtualUI Server Manager GUI.</td>
</tr>
</tbody>
</table>

© 2015, Cybele Software, Inc.
9.3.2.1.3.1 TServer.Load Method

Loads all the configuration entries and profiles from file. It is automatically called by constructor.

**Pascal**

```pascal
procedure Load; safecall;
```

**Group**

TServer Methods

9.3.2.1.3.2 TServer.Save Method

Saves the entire configuration parameters and profiles.

**Pascal**

```pascal
procedure Save; safecall;
```

**Group**

TServer Methods

9.3.2.1.3.3 TServer.HideSection Method

Hides a configuration section in the VirtualUI Server Manager GUI.

**Pascal**

```pascal
procedure HideSection(section: ServerSection); safecall;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
</table>
| section: ServerSection | The Server configuration section to hide to user. Use one of the following constants:  
  - SRVSEC_GENERAL: Hides the General tab, that contains the Binding configuration.  
  - SRVSEC_RDS: Hides the tab with the Remote Desktop Services account configuration.  
  - SRVSEC_APPLICATIONS: Hides the list of applications.  
  - SRVSEC_LICENSES: Hides the tab with License information. |

**Description**

**Group**

TServer Methods
9.3.2.1.3.4 TServer.ShowSection Method

Makes visible a configuration section in the VirtualUI Server Manager GUI.

**Pascal**

```pascal
procedure ShowSection(section: ServerSection); safecall;
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>section: ServerSection</td>
<td>The Server configuration section to show to user. Use one of the following constants:</td>
</tr>
<tr>
<td></td>
<td>• SRVSEC_GENERAL: Shows the General tab, that contains the Binding configuration.</td>
</tr>
<tr>
<td></td>
<td>• SRVSEC_RDS: Shows the tab with the Remote Desktop Services account configuration.</td>
</tr>
<tr>
<td></td>
<td>• SRVSEC_APPLICATIONS: Shows the list of applications.</td>
</tr>
<tr>
<td></td>
<td>• SRVSEC_LICENSES: Shows the tab with License information.</td>
</tr>
</tbody>
</table>

### Description

#### Group

TServer Methods

9.3.2.1.4 TServer Properties

The properties of the TServer class are listed here.

### Class

TServer Class

#### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binding</td>
<td>Returns the Server's Binding configuration.</td>
</tr>
<tr>
<td>Certificate</td>
<td>Returns the Server's certificate configuration for SSL protocol.</td>
</tr>
<tr>
<td>RDSAccounts</td>
<td>Returns the list of Remote Desktop Services accounts.</td>
</tr>
<tr>
<td>Profiles</td>
<td>Returns the profiles list.</td>
</tr>
<tr>
<td>License</td>
<td>Returns the current Server's licence.</td>
</tr>
</tbody>
</table>

9.3.2.1.4.1 TServer.Binding Property

Returns the Server's Binding configuration.
Pascal
property Binding: IBinding;

See Also
IBinding interface

Group
TServer Properties

9.3.2.1.4.2 TServer.Certificate Property

Returns the Server's certificate configuration for SSL protocol.

Pascal
property Certificate: ICertificate;

See Also
ICertificate interface

Group
TServer Properties

9.3.2.1.4.3 TServer.RDSAccounts Property

Returns the list of Remote Desktop Services accounts.

Pascal
property RDSAccounts: IRDSAccounts;

See Also
IRDSAccounts interface

Group
TServer Properties

9.3.2.1.4.4 TServer.Profiles Property

Returns the profiles list.

Pascal
property Profiles: IProfiles;

See Also
IProfiles interface

Group

TServer Properties

9.3.2.1.4.5 TServer.License Property

Returns the current Server's licence.

Pascal

property License: ILicense;

See Also

ILicense interface

Group

TServer Properties

9.3.2.2 VirtualUI_Settings.Base64ToIcon Function

Converts the IProfile.IconData (base64 string) to a PNG image.

Pascal

function Base64ToIcon(AData: String): TPngImage;

File

VirtualUI_Settings

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>The image encoded in base64.</td>
</tr>
</tbody>
</table>

Namespace

VirtualUI_Settings Unit

9.3.2.3 VirtualUI_Settings.IconToBase64 Function

Converts a PNG image to be stored in IProfile.IconData (as base64 string).

Pascal

function IconToBase64(png: TPngImage): string;

File
VirtualUI_Settings

## Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>png: TPngImage</td>
<td>The image to be encoded in base64.</td>
</tr>
</tbody>
</table>

## Namespace

VirtualUI_Settings Unit

### 9.4 C++ Classes

The following table lists classes in this documentation.

## Description

### Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CClientSettings</td>
<td>Allows to set some client settings.</td>
</tr>
<tr>
<td>CBrowserInfo</td>
<td>Contains information regarding the end-user’s screen, web browser, the window containing VirtualUI Viewer and VirtualUI Viewer itself. The VirtualUI Viewer runs inside an HTML DIV element contained in a frame of browser window on the end-user's application page.</td>
</tr>
<tr>
<td>CDevServer</td>
<td>Contains properties to manage the VirtualUI Development Server as well as the access from the developer's web browser.</td>
</tr>
<tr>
<td>JSObject</td>
<td>Represents a custom remotable object.</td>
</tr>
<tr>
<td>JSCallback</td>
<td>This is class JSCallback.</td>
</tr>
<tr>
<td>JSBinding</td>
<td>This is class JSBinding.</td>
</tr>
</tbody>
</table>

### Topics

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualUI</td>
<td></td>
</tr>
</tbody>
</table>

### 9.4.1 CClientSettings Class

Allows to set some client settings.

### Class Hierarchy

```
IClientSettings → CClientSettings
```

© 2015, Cybele Software, Inc.
class CClientSettings : public IClientSettings;

File
Thinfinity.VirtualUI.h

Group
C++ Classes

9.4.1.1 CClientSettings::CClientSettings Constructor

C++
CClientSettings(IVirtualUI * virtualUI);

Description
This is CClientSettings, a member of class CClientSettings.

Class
CClientSettings Class

9.4.1.2 CClientSettings::~CClientSettings Destructor

C++
~CClientSettings();

Description
This is ~CClientSettings, a member of class CClientSettings.

Class
CClientSettings Class

9.4.1.3 CClientSettings Methods

The methods of the CClientSettings class are listed here.

Class
CClientSettings Class

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CursorVisible</td>
<td>This is the overview for the CursorVisible method overload.</td>
</tr>
<tr>
<td>MouseMoveGestAction</td>
<td>This is the overview for the MouseMoveGestAction method</td>
</tr>
</tbody>
</table>
9.4.1.3.1 CursorVisible Method

This is the overview for the CursorVisible method overload.

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CClientSettings::CursorVisible()</td>
<td>Gets the the mouse pointer state.</td>
</tr>
<tr>
<td>CClientSettings::CursorVisible(bool)</td>
<td>Hides/shows the mouse pointer.</td>
</tr>
</tbody>
</table>

### Group

CClientSettings Methods

9.4.1.3.1.1 CClientSettings::CursorVisible Method ()

Gets the the mouse pointer state.

```
C++
bool CursorVisible();
```

### Group

CursorVisible Method

9.4.1.3.1.2 CClientSettings::CursorVisible Method (bool)

Hides/shows the mouse pointer.

```
C++
void CursorVisible(bool value);
```
9.4.1.3.2  MouseMoveGestAction Method

This is the overview for the MouseMoveGestAction method overload.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CClientSettings::MouseMoveGestAction ()</td>
<td>Gets the MouseMoveGestureAction value</td>
</tr>
<tr>
<td>CClientSettings::MouseMoveGestAction (MouseMoveGestureAction)</td>
<td>Specifies whether the &quot;mouse move&quot; simulation on a touch device is interpreted as a mouse move or as a mouse wheel.</td>
</tr>
</tbody>
</table>

9.4.1.3.2.1  CClientSettings::MouseMoveGestAction Method ()

Gets the MouseMoveGestureAction value

**C++**

MouseMoveGestureAction MouseMoveGestAction();

9.4.1.3.2.2  CClientSettings::MouseMoveGestAction Method (MouseMoveGestureAction)

 Specifies whether the "mouse move" simulation on a touch device is interpreted as a mouse move or as a mouse wheel.

**C++**

void MouseMoveGestAction(MouseMoveGestureAction value);

9.4.1.3.3  MouseMoveGestStyle Method

This is the overview for the MouseMoveGestStyle method overload.
### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CClientSettings::MouseMoveGestureStyle()</td>
<td>Gets the MouseMoveGestureStyle value</td>
</tr>
<tr>
<td>CClientSettings::MouseMoveGestureStyle(MouseMoveGestureStyle)</td>
<td>Valid for touch devices. Specifies whether the mouse pointer is shown and acts on the exact spot of the finger touch (absolute) or its position is managed relatively to the movement of the finger touch (relative).</td>
</tr>
</tbody>
</table>

### Group

#### CClientSettings Methods

#### 9.4.1.3.3.1 CClientSettings::MouseMoveGestStyle Method ()

Gets the MouseMoveGestureStyle value

**C++**

```cpp
MouseMoveGestureStyle MouseMoveGestStyle();
```

#### Group

MouseMoveGestStyle Method

#### 9.4.1.3.3.2 CClientSettings::MouseMoveGestStyle Method (MouseMoveGestureStyle)

Valid for touch devices. Specifies whether the mouse pointer is shown and acts on the exact spot of the finger touch (absolute) or its position is managed relatively to the movement of the finger touch (relative).

**C++**

```cpp
void MoveMouseGestStyle(MouseMoveGestureStyle value);
```

#### Group

MouseMoveGestStyle Method

#### 9.4.1.3.4 CClientSettings::get_MouseMoveGestureStyle Method

**C++**

```cpp
__stdcall HRESULT get_MouseMoveGestureStyle(enum MouseMoveGestureStyle * Value);
```

### Description

This is get_MouseMoveGestureStyle, a member of class CClientSettings.

#### Group

CClientSettings Methods
9.4.1.3.5  CClientSettings::put_MouseMoveGestureStyle Method

```cpp
__stdcall HRESULT put_MouseMoveGestureStyle(enum MouseMoveGestureStyle Value);
```

**Description**
This is put_MouseMoveGestureStyle, a member of class CClientSettings.

**Group**
CClientSettings Methods

9.4.1.3.6  CClientSettings::get_MouseMoveGestureAction Method

```cpp
__stdcall HRESULT get_MouseMoveGestureAction(enum MouseMoveGestureAction * Value);
```

**Description**
This is get_MouseMoveGestureAction, a member of class CClientSettings.

**Group**
CClientSettings Methods

9.4.1.3.7  CClientSettings::put_MouseMoveGestureAction Method

```cpp
__stdcall HRESULT put_MouseMoveGestureAction(enum MouseMoveGestureAction Value);
```

**Description**
This is put_MouseMoveGestureAction, a member of class CClientSettings.

**Group**
CClientSettings Methods

9.4.1.3.8  CClientSettings::get_CursorVisible Method

```cpp
__stdcall HRESULT get_CursorVisible(VARIANT_BOOL * Value);
```

**Description**
This is get_CursorVisible, a member of class CClientSettings.

**Group**
CClientSettings Methods
9.4.1.3.9  CClientSettings::put_CursorVisible Method

```cpp
__stdcall HRESULT put_CursorVisible(VARIANT_BOOL Value);
```

**Description**
This is put_CursorVisible, a member of class CClientSettings.

**Group**
CClientSettings Methods

9.4.2  CBrowserInfo Class

Contains information regarding the end-user's screen, web browser, the window containing VirtualUI Viewer and VirtualUI Viewer itself. The VirtualUI Viewer runs inside an HTML DIV element contained in a frame of browser window on the end-user's application page.

**Class Hierarchy**
```
 class CBrowserInfo : public IBrowserInfo;
```

**File**
Tinfinity.VirtualUI.h

**Group**
C++ Classes

9.4.2.1  CBrowserInfo::CBrowserInfo Constructor

```cpp
CBrowserInfo(IVirtualUI * virtualUI);
```

**Description**
This is CBrowserInfo, a member of class CBrowserInfo.

**Class**
CBrowserInfo Class
9.4.2.2 CBrowserInfo::~CBrowserInfo Destructor

```cpp
~CBrowserInfo();
```

**Description**

This is ~CBrowserInfo, a member of class CBrowserInfo.

**Class**

CBrowserInfo Class

9.4.2.3 CBrowserInfo Methods

The methods of the CBrowserInfo class are listed here.

**Class**

CBrowserInfo Class

**Public Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewHeight</td>
<td>This is the overview for the ViewHeight method overload.</td>
</tr>
<tr>
<td>ViewWidth</td>
<td>This is the overview for the ViewWidth method overload.</td>
</tr>
<tr>
<td>get_ViewWidth</td>
<td>This is get_ViewWidth, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>put_ViewWidth</td>
<td>This is put_ViewWidth, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>get_ViewHeight</td>
<td>This is get_ViewHeight, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>put_ViewHeight</td>
<td>This is put_ViewHeight, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>get_BrowserWidth</td>
<td>This is get_BrowserWidth, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>get_BrowserHeight</td>
<td>This is get_BrowserHeight, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>get_ScreenWidth</td>
<td>This is get_ScreenWidth, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>get_ScreenHeight</td>
<td>This is get_ScreenHeight, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>get_Username</td>
<td>This is get_Username, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>get_IPAddress</td>
<td>This is get_IPAddress, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>get_UserAgent</td>
<td>This is get_UserAgent, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>get_ScreenResolution</td>
<td>This is get_ScreenResolution, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>get_Orientation</td>
<td>This is get_Orientation, a member of class CBrowserInfo.</td>
</tr>
</tbody>
</table>
get_UniqueBrowserId

This is get_UniqueBrowserId, a member of class CBrowserInfo.

GetCookie

This is GetCookie, a member of class CBrowserInfo.

SetCookie

This is SetCookie, a member of class CBrowserInfo.

get_Location

This is get_Location, a member of class CBrowserInfo.

BrowserWidth

Returns the width of the HTML element containing the VirtualUI Viewer.

BrowserHeight

Returns the height of the HTML element containing the VirtualUI Viewer.

ScreenWidth

Returns the width of the end-user’s monitor screen.

ScreenHeight

Returns the height of the end-user’s monitor screen.

Username

Returns the logged-on Username.

IPAddress

Returns the client’s IP address.

UserAgent

Returns the browser’s User Agent string.

UniqueBrowserId

UniqueBrowserId identifies an instance of a Web Browser. Each time an end-user opens the application from a different browser window, this ID will have a different value.

Location

Returns the URL of the current application.

ScreenResolution

Returns the application screen resolution defined in the application profile.

Orientation

Returns the browser’s orientation.

9.4.2.3.1 ViewHeight Method

This is the overview for the ViewHeight method overload.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBrowserInfo::ViewHeight ()</td>
<td>Returns the height of the VirtualUI Viewer.</td>
</tr>
<tr>
<td>CBrowserInfo::ViewHeight (int)</td>
<td>Sets the height of the VirtualUI Viewer.</td>
</tr>
</tbody>
</table>

Group

CBrowserInfo Methods

9.4.2.3.1.1 CBrowserInfo::ViewHeight Method ()

Returns the height of the VirtualUI Viewer.
int ViewHeight();

Group

ViewHeight Method

9.4.2.3.1.2 CBrowserInfo::ViewHeight Method (int)

Sets the height of the VirtualUI Viewer.

C++

void ViewHeight(int value);

Group

ViewHeight Method

9.4.2.3.2 ViewWidth Method

This is the overview for the ViewWidth method overload.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBrowserInfo::ViewWidth()</td>
<td>Returns the width of the VirtualUI Viewer.</td>
</tr>
<tr>
<td>CBrowserInfo::ViewWidth(int)</td>
<td>Sets the width of the VirtualUI Viewer.</td>
</tr>
</tbody>
</table>

Group

CBrowserInfo Methods

9.4.2.3.2.1 CBrowserInfo::ViewWidth Method ()

Returns the width of the VirtualUI Viewer.

C++

int ViewWidth();

Group

ViewWidth Method

9.4.2.3.2.2 CBrowserInfo::ViewWidth Method (int)

Sets the width of the VirtualUI Viewer.

C++

void ViewWidth(int value);
9.4.2.3.3 CBrowserInfo::get_ViewWidth Method

C++
__stdcall HRESULT get_ViewWidth(long * Value);

Description
This is get_ViewWidth, a member of class CBrowserInfo.

9.4.2.3.4 CBrowserInfo::put_ViewWidth Method

C++
__stdcall HRESULT put_ViewWidth(long Value);

Description
This is put_ViewWidth, a member of class CBrowserInfo.

9.4.2.3.5 CBrowserInfo::get_ViewHeight Method

C++
__stdcall HRESULT get_ViewHeight(long * Value);

Description
This is get_ViewHeight, a member of class CBrowserInfo.

9.4.2.3.6 CBrowserInfo::put_ViewHeight Method

C++
__stdcall HRESULT put_ViewHeight(long Value);

Description
This is put_ViewHeight, a member of class CBrowserInfo.

Group

CBrowserInfo Methods

9.4.2.3.7 CBrowserInfo::get_BrowserWidth Method

```cpp
__stdcall HRESULT get_BrowserWidth(long * Value);
```

Description

This is get_BrowserWidth, a member of class CBrowserInfo.

Group

CBrowserInfo Methods

9.4.2.3.8 CBrowserInfo::get_BrowserHeight Method

```cpp
__stdcall HRESULT get_BrowserHeight(long * Value);
```

Description

This is get_BrowserHeight, a member of class CBrowserInfo.

Group

CBrowserInfo Methods

9.4.2.3.9 CBrowserInfo::get_ScreenWidth Method

```cpp
__stdcall HRESULT get_ScreenWidth(long * Value);
```

Description

This is get_ScreenWidth, a member of class CBrowserInfo.

Group

CBrowserInfo Methods

9.4.2.3.10 CBrowserInfo::get_ScreenHeight Method

```cpp
__stdcall HRESULT get_ScreenHeight(long * Value);
```
Description
This is get_ScreenHeight, a member of class CBrowserInfo.

Group
CBrowserInfo Methods

9.4.2.3.11 CBrowserInfo::get_Username Method

C++
__stdcall HRESULT get_Username(BSTR * Value);

Description
This is get_Username, a member of class CBrowserInfo.

Group
CBrowserInfo Methods

9.4.2.3.12 CBrowserInfo::get_IPAddress Method

C++
__stdcall HRESULT get_IPAddress(BSTR * Value);

Description
This is get_IPAddress, a member of class CBrowserInfo.

Group
CBrowserInfo Methods

9.4.2.3.13 CBrowserInfo::get_UserAgent Method

C++
__stdcall HRESULT get_UserAgent(BSTR * Value);

Description
This is get_UserAgent, a member of class CBrowserInfo.

Group
CBrowserInfo Methods
9.4.2.3.14 CBrowserInfo::get_ScreenResolution Method

```cpp
__stdcall HRESULT get_ScreenResolution(long * Value);
```

**Description**
This is get_ScreenResolution, a member of class CBrowserInfo.

**Group**
CBrowserInfo Methods

9.4.2.3.15 CBrowserInfo::get_Orientation Method

```cpp
__stdcall HRESULT get_Orientation(enum Orientation * Value);
```

**Description**
This is get_Orientation, a member of class CBrowserInfo.

**Group**
CBrowserInfo Methods

9.4.2.3.16 CBrowserInfo::get_UniqueBrowserId Method

```cpp
__stdcall HRESULT get_UniqueBrowserId(BSTR * Value);
```

**Description**
This is get_UniqueBrowserId, a member of class CBrowserInfo.

**Group**
CBrowserInfo Methods

9.4.2.3.17 CBrowserInfo::GetCookie Method

```cpp
__stdcall HRESULT GetCookie(BSTR Name, BSTR * Value);
```

**Description**
This is GetCookie, a member of class CBrowserInfo.

**Group**
CBrowserInfo Methods
CBrowserInfo Methods

9.4.2.3.18 CBrowserInfo::SetCookie Method

```cpp
__stdcall HRESULT SetCookie(BSTR Name, BSTR Value, BSTR Expires);
```

- **Description**
  This is SetCookie, a member of class CBrowserInfo.

- **Group**
  CBrowserInfo Methods

9.4.2.3.19 CBrowserInfo::get_Location Method

```cpp
__stdcall HRESULT get_Location(BSTR * Value);
```

- **Description**
  This is get_Location, a member of class CBrowserInfo.

- **Group**
  CBrowserInfo Methods

9.4.2.3.20 CBrowserInfo::BrowserWidth Method

```cpp
int BrowserWidth();
```

- **Description**
  Returns the width of the HTML element containing the VirtualUI Viewer.

- **Group**
  CBrowserInfo Methods

9.4.2.3.21 CBrowserInfo::BrowserHeight Method

Returns the height of the HTML element containing the VirtualUI Viewer.

```cpp
int BrowserHeight();
```

- **Group**
  CBrowserInfo Methods
CBrowserInfo Methods

9.4.2.3.22 CBrowserInfo::ScreenWidth Method

Returns the width of the end-user's monitor screen.

```cpp
int ScreenWidth();
```

9.4.2.3.23 CBrowserInfo::ScreenHeight Method

Returns the height of the end-user's monitor screen.

```cpp
int ScreenHeight();
```

9.4.2.3.24 CBrowserInfo::Username Method

Returns the logged-on Username.

```cpp
std::wstring Username();
```

9.4.2.3.25 CBrowserInfo::IPAddress Method

Returns the client's IP address.

```cpp
std::wstring IPAddress();
```

9.4.2.3.26 CBrowserInfo::UserAgent Method

Returns the browser's User Agent string.
9.4.2.3.27  CBrowserInfo::UniqueBrowserId Method

UniqueBrowserId identifies an instance of a Web Browser. Each time an end-user opens the application from a different browser window, this ID will have a different value.

9.4.2.3.28  CBrowserInfo::Location Method

Returns the URL of the current application.

9.4.2.3.29  CBrowserInfo::ScreenResolution Method

Returns the application screen resolution defined in the application profile.

9.4.2.3.30  CBrowserInfo::Orientation Method

Returns the browser's orientation.
CBrowserInfo Methods

9.4.3 CDevServer Class

Contains properties to manage the VirtualUI Development Server as well as the access from the developer's web browser.

Class Hierarchy

```
  IDevServer
     `-- CDevServer
```

C++

```cpp
class CDevServer : public IDevServer;
```

File

Thinfinity.VirtualUI.h

Group

C++ Classes

9.4.3.1 CDevServer::CDevServer Constructor

C++

```cpp
CDevServer(IVirtualUI * virtualUI);
```

Description

This is CDevServer, a member of class CDevServer.

Class

CDevServer Class

9.4.3.2 CDevServer::~CDevServer Destructor

C++

```cpp
~CDevServer();
```

Description

This is ~CDevServer, a member of class CDevServer.

Class

CDevServer Class
9.4.3.3  CDevServer Methods

The methods of the CDevServer class are listed here.

Class
CDevServer Class

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>This is the overview for the Enabled method overload.</td>
</tr>
<tr>
<td>Port</td>
<td>This is the overview for the Port method overload.</td>
</tr>
<tr>
<td>get_Enabled</td>
<td>This is get_Enabled, a member of class CDevServer.</td>
</tr>
<tr>
<td>put_Enabled</td>
<td>This is put_Enabled, a member of class CDevServer.</td>
</tr>
<tr>
<td>get_Port</td>
<td>This is get_Port, a member of class CDevServer.</td>
</tr>
<tr>
<td>put_Port</td>
<td>This is put_Port, a member of class CDevServer.</td>
</tr>
<tr>
<td>get_StartBrowser</td>
<td>This is get_StartBrowser, a member of class CDevServer.</td>
</tr>
<tr>
<td>put_StartBrowser</td>
<td>This is put_StartBrowser, a member of class CDevServer.</td>
</tr>
<tr>
<td>StartBrowser</td>
<td>Instructs VirtualUI whether start or not the local web browser upon VirtualUI activation.</td>
</tr>
</tbody>
</table>

9.4.3.3.1  Enabled Method

This is the overview for the Enabled method overload.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDevServer::Enabled ()</td>
<td>Gets the Development Server state.</td>
</tr>
<tr>
<td>CDevServer::Enabled (bool)</td>
<td>Enables/disables the Development Server.</td>
</tr>
</tbody>
</table>

Group
CDevServer Methods

9.4.3.3.1.1  CDevServer::Enabled Method ()

Gets the Development Server state.

C++

```cpp
bool Enabled();
```
9.4.3.3.1.2 CDevServer::Enabled Method (bool)

Enables/disables the Development Server.

**C++**

```cpp
void Enabled(bool value);
```

9.4.3.3.2 Port Method

This is the overview for the Port method overload.

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDevServer::Port ()</td>
<td>Gets the Development Server's TCP/IP listening port.</td>
</tr>
<tr>
<td>CDevServer::Port (int)</td>
<td>Sets the Development Server's TCP/IP listening port.</td>
</tr>
</tbody>
</table>

9.4.3.3.2.1 CDevServer::Port Method ()

Gets the Development Server's TCP/IP listening port.

**C++**

```cpp
int Port();
```

9.4.3.3.2.2 CDevServer::Port Method (int)

Sets the Development Server's TCP/IP listening port.

**C++**

```cpp
void Port(int value);
```
9.4.3.3.3  CDevServer::get_Enabled Method

```
C++
__stdcall HRESULT get_Enabled(VARIANT_BOOL * Value);
```

**Description**

This is get_Enabled, a member of class CDevServer.

**Group**

CDevServer Methods

9.4.3.3.4  CDevServer::put_Enabled Method

```
C++
__stdcall HRESULT put_Enabled(VARIANT_BOOL Value);
```

**Description**

This is put_Enabled, a member of class CDevServer.

**Group**

CDevServer Methods

9.4.3.3.5  CDevServer::get_Port Method

```
C++
__stdcall HRESULT get_Port(long * Value);
```

**Description**

This is get_Port, a member of class CDevServer.

**Group**

CDevServer Methods

9.4.3.3.6  CDevServer::put_Port Method

```
C++
__stdcall HRESULT put_Port(long Value);
```

**Description**

This is put_Port, a member of class CDevServer.
Group
CDevServer Methods

9.4.3.3.7 CDevServer::get_StartBrowser Method

C++
__stdcall HRESULT get_StartBrowser(VARIANT_BOOL * Value);

Description
This is get_StartBrowser, a member of class CDevServer.

Group
CDevServer Methods

9.4.3.8 CDevServer::put_StartBrowser Method

C++
__stdcall HRESULT put_StartBrowser(VARIANT_BOOL Value);

Description
This is put_StartBrowser, a member of class CDevServer.

Group
CDevServer Methods

9.4.3.9 CDevServer::StartBrowser Method

Instructs VirtualUI whether start or not the local web browser upon VirtualUI activation.

C++
void StartBrowser(bool value);

Group
CDevServer Methods

9.4  JSObject Class

Represents a custom remotable object.

Class Hierarchy
C++

`class JSObject : public VirtualUILibrary, private IJSObject;`

- **File**
  Thinfinity.VirtualUI.h

- **Remarks**
  JSObject allows you to define an object model that is mirrored on the client side, and allows for an easy, powerful and straight-forward way to connect the web browser client application and the remoted Windows application.

  JSObject can contain properties (IJSProperties), methods (IJSMethods), events (IJSEvents) and children objects. Changes to properties values are propagated in from server to client and viceversa, keeping the data synchronized.

  JSObject is defined as a model seen from the client perspective. A method (IJSMethod) is called on the client side and executed on the server side. An event (IJSEvent) is called on the server side and raised on the client side.

- **Group**
  C++ Classes

### 9.4.4.1 JSObject::JSObject Constructor

**C++**

```cpp
JSObject(std::wstring id);
```

- **Description**
  This is JSObject, a member of class JSObject.

### 9.4.4.2 JSObject::~JSObject Destructor

**C++**

```cpp
~JSObject();
```

- **Description**
  This is ~JSObject, a member of class JSObject.
9.4.4.3  JSObject Methods

The methods of the JSObject class are listed here.

### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FireEvent</td>
<td>This is the overview for the FireEvent method overload.</td>
</tr>
<tr>
<td>Id</td>
<td>This is the overview for the Id method overload.</td>
</tr>
<tr>
<td>QueryInterface</td>
<td>This is QueryInterface, a member of class JSObject.</td>
</tr>
<tr>
<td>get_Id</td>
<td>This is get_Id, a member of class JSObject.</td>
</tr>
<tr>
<td>put_Id</td>
<td>This is put_Id, a member of class JSObject.</td>
</tr>
<tr>
<td>get_Properties</td>
<td>This is get_Properties, a member of class JSObject.</td>
</tr>
<tr>
<td>get_Methods</td>
<td>This is get_Methods, a member of class JSObject.</td>
</tr>
<tr>
<td>get_Events</td>
<td>This is get_Events, a member of class JSObject.</td>
</tr>
<tr>
<td>get_Objects</td>
<td>This is get_Objects, a member of class JSObject.</td>
</tr>
<tr>
<td>ApplyChanges</td>
<td>When this method called, all properties getters are internally called looking for changes. Any change to the property value is sent to the client.</td>
</tr>
<tr>
<td>ApplyModel</td>
<td>Propagates the whole JSObject definition to the javascript client.</td>
</tr>
<tr>
<td>Properties</td>
<td>List containing all properties of this object.</td>
</tr>
<tr>
<td>Methods</td>
<td>List containing all methods of this object.</td>
</tr>
<tr>
<td>Events</td>
<td>List containing all events of this object.</td>
</tr>
<tr>
<td>Objects</td>
<td>List containing all events of this object.</td>
</tr>
</tbody>
</table>

9.4.4.3.1  FireEvent Method

This is the overview for the FireEvent method overload.

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Group
**JSObject Methods**

#### 9.4.4.3.1.1 JSObject::FireEvent Method (BSTR, IJSArguments *)

**C++**

```cpp
__stdcall HRESULT FireEvent(BSTR Name, IJSArguments * Arguments);
```

**Description**

This is `FireEvent`, a member of class `JSObject`.

#### Group
**FireEvent Method**

#### 9.4.4.3.1.2 JSObject::FireEvent Method (std::wstring, IJSArguments*)

**C++**

```cpp
void FireEvent(std::wstring Name, IJSArguments* Arguments);
```

**Description**

This is `FireEvent`, a member of class `JSObject`.

#### Group
**FireEvent Method**

#### 9.4.4.3.2 Id Method

This is the overview for the `Id` method overload.

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSObject::Id ()</td>
<td>Identifier of the object. It must be unique among siblings objects.</td>
</tr>
<tr>
<td>JSObject::Id (std::wstring)</td>
<td>Identifier of the object. It must be unique among siblings objects.</td>
</tr>
</tbody>
</table>

#### Group
**JSObject Methods**
9.4.4.3.2.1 JSObject::Id Method ()
Identifier of the object. It must be unique among siblings objects.

```cpp
std::wstring Id();
void Id(std::wstring value);
```

9.4.4.3.3 JSObject::QueryInterface Method

```cpp
__stdcall HRESULT QueryInterface(REFIID riid, void ** ppvObject);
```

**Description**
This is QueryInterface, a member of class JSObject.

9.4.4.3.4 JSObject::get_Id Method

```cpp
__stdcall HRESULT get_Id(BSTR * Value);
```

**Description**
This is get_Id, a member of class JSObject.

9.4.4.3.5 JSObject::put_Id Method

```cpp
__stdcall HRESULT put_Id(BSTR Value);
```

**Description**
This is put_Id, a member of class JSObject.
9.4.4.3.6 JSObject::get_Properties Method

```cpp
__stdcall HRESULT get_Properties(IJSProperties ** Value);
```

**Description**
This is get_Properties, a member of class JSObject.

**Group**
JSObject Methods

9.4.4.3.7 JSObject::get_Methods Method

```cpp
__stdcall HRESULT get_Methods(IJSMethods ** Value);
```

**Description**
This is get_Methods, a member of class JSObject.

**Group**
JSObject Methods

9.4.4.3.8 JSObject::get_Events Method

```cpp
__stdcall HRESULT get_Events(IJSEvents ** Value);
```

**Description**
This is get_Events, a member of class JSObject.

**Group**
JSObject Methods

9.4.4.3.9 JSObject::get_Objects Method

```cpp
__stdcall HRESULT get_Objects(IJSObjects ** Value);
```

**Description**
This is get_Objects, a member of class JSObject.

**Group**
JSObject Methods
**JSObject Methods**

### 9.4.4.3.10 JSObject::ApplyChanges Method

When this method called, all properties getters are internally called looking for changes. Any change to the property value is sent to the client.

\[ C++ \]
__stdcall HRESULT ApplyChanges();

**Group**

### 9.4.4.3.11 JSObject::ApplyModel Method

Propagates the whole JSObject definition to the javascript client.

\[ C++ \]
__stdcall HRESULT ApplyModel();

**Group**

### 9.4.4.3.12 JSObject::Properties Method

List containing all properties of this object.

\[ C++ \]
CJSProperties* Properties();

**Group**

### 9.4.4.3.13 JSObject::Methods Method

List containing all methods of this object.

\[ C++ \]
CJSMethods* Methods();

**Group**

### 9.4.4.3.14 JSObject::Events Method

List containing all events of this object.
C++
CJSEvents* Events();

Group
JSObject Methods

9.4.4.3.15 JSObject::Objects Method

List containing all events of this object.

C++
CJSObjects* Objects();

Group
JSObject Methods

9.4.5 JSCallback Class

Class Hierarchy

![Class Hierarchy Diagram]

C++
class JSCallback : private IJSCallback;

File
Thinfinity.VirtualUI.h

Description
This is class JSCallback.

Group
C++ Classes

9.4.5.1 JSCallback::JSCallback Constructor

C++
JSCallback(JSCallbackProc);

Description
This is JSCallback, a member of class JSCallback.
JSCallback Class

9.4.5.2 JSCallback Methods

The methods of the JSCallback class are listed here.

- **Class**
  JSCallback Class

- **Public Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callback</td>
<td>This is Callback, a member of class JSCallback.</td>
</tr>
</tbody>
</table>

9.4.5.2.1 JSCallback::Callback Method

**C++**

```cpp
__stdcall HRESULT Callback(IJSObject* Parent, IJSMethod* Method);
```

- **Description**

  This is Callback, a member of class JSCallback.

- **Group**
  JSCallback Methods

9.4.6 JSBinding Class

- **Class Hierarchy**

  ```cpp
  class JSBinding : private IJSBinding;
  ```

- **File**

  Thinfinity.VirtualUI.h

- **Description**

  This is class JSBinding.

- **Group**
  C++ Classes
9.4.6.1 JSBinding::JSBinding Constructor

```cpp
JSBinding{JSPropertySet Proc};
```

**Description**
This is JSBinding, a member of class JSBinding.

**Class**
JSBinding Class

9.4.6.2 JSBinding Methods

The methods of the JSBinding class are listed here.

**Class**
JSBinding Class

**Public Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>This is Set, a member of class JSBinding.</td>
</tr>
</tbody>
</table>

9.4.6.2.1 JSBinding::Set Method

```cpp
__stdcall HRESULT Set(IJSObject* Parent, IJSProperty* Prop);
```

**Description**
This is Set, a member of class JSBinding.

**Group**
JSBinding Methods

9.4.7 VirtualUI

**Group**
C++ Classes
9.5 VirtualUI.sdk.min.js

### Files

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>thinfinity.virtualui</td>
<td>Thinfinity.<strong>VirtualUI</strong> is a Javascript pseudo-class that offers a single-line-of-code user interface remoting solution for self-developed applications, making it possible to deliver them as dual-platform Windows/HTML5 apps. Regarding its full web integration capabilities, Thinfinity <strong>VirtualUI</strong> takes advantage of the newly developed Javascript Remote Objects (jsRO) framework to facilitate two-way data integration between the application and the browser. It empowers developers who are confronted with a need for deep modernization of existing Windows-based apps with a dual-approach tool: instant GUI remoting and full web integration.</td>
</tr>
<tr>
<td>thinfinity.jsro</td>
<td>Thinfinity JsRO is a Javascript pseudo-class for GUI remoting and full web integration. It allows you to create remotable objects, and their properties, methods and events are mirrored to the web as native javascript objects. There must be a unique HOST to communicate with, and there can be more than one REMOTE for the same HOST. The HOST will keep all REMOTES synchronized, even when a REMOTE changes a value locally the other REMOTES will receive a notification.</td>
</tr>
</tbody>
</table>

#### 9.5.1 Thinfinity.VirtualUI

Thinfinity.**VirtualUI** is a Javascript pseudo-class that offers a single-line-of-code user interface remoting solution for self-developed applications, making it possible to deliver them as dual-platform Windows/HTML5 apps.

Regarding its full web integration capabilities, Thinfinity **VirtualUI** takes advantage of the newly developed Javascript Remote Objects (jsRO) framework to facilitate two-way data integration between the application and the browser.

It empowers developers who are confronted with a need for deep modernization of existing Windows-based apps with a dual-approach tool: instant GUI remoting and full web integration.

**JavaScript**

```javascript
var virtualUI = new Thinfinity.VirtualUI(); // has no arguments
```

#### Returns

Thinfinity.**VirtualUI** Object.

#### Version

20141216.1

#### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onLoading</td>
<td>Occurs when 'loading' is triggered. The event will be fired once the client and the server are connected to each other.</td>
</tr>
</tbody>
</table>
onShow
Occurs when 'show' is triggered. The event will be fired once the web-app is ready to be shown.

onClose
Occurs when 'close' is triggered. The event will be fired once the app is closed.

onError
Occurs when 'error' is triggered.

Group
VirtualUI.sdk.min.js

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devMode</td>
<td>Retrieves information about the current mode. In the developer environment only one application can run at the same time. By default the assigned port is 6080, if it's available.</td>
</tr>
</tbody>
</table>

9.5.1.1 Events

9.5.1.1.1 onLoading

JavaScript
virtualUI.onLoading = function(){
  //...TODO
};

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handler</td>
<td>callback as function.</td>
</tr>
</tbody>
</table>

Description
Occurs when 'loading' is triggered. The event will be fired once the client and the server are connected to each other.

File

Thinfinity.VirtualUI

9.5.1.1.2 onShow

JavaScript
virtualUI.onShow = function(){
  //...TODO
};

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handler</td>
<td>callback as function.</td>
</tr>
</tbody>
</table>
Description

Occurs when 'show' is triggered. The event will be fired once the web-app is ready to be shown.

File

Thinfinity.VirtualUI

9.5.1.1.3 onClose

JavaScript

virtualUI.onClose = function(){
  //...TODO
};

Description

Occurs when 'close' is triggered. The event will be fired once the app is closed.

File

Thinfinity.VirtualUI

9.5.1.1.4 onError

JavaScript

virtualUI.onError = function(message){
  //...TODO
};

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handler</td>
<td>callback as function. The event will be fired once the app generates an error at run-time.</td>
</tr>
</tbody>
</table>

Returns

message as string

Description

Occurs when 'error' is triggered.

File

Thinfinity.VirtualUI
9.5.1.2 Properties

9.5.1.2.1 devMode

JavaScript

```javascript
virtualUI.devMode;
if (virtualUI.devMode === true){
  //...TODO
}
```

- **Returns**

  Returns Boolean, depending on the mode value.

- **Description**

  Retrieves information about the current mode. In the developer environment only one application can run at the same time. By default the assigned port is 6080, if it's available.

- **File**

  `Thinfinity.VirtualUI`

9.5.2 Thinfinity.JsRO

Thinfinity.JsRO is a Javascript pseudo-class for GUI remoting and full web integration. It allows you to create remotable objects, and their properties, methods and events are mirrored to the web as native javascript objects. There must be a unique HOST to communicate with, and there can be more than one REMOTE for the same HOST. The HOST will keep all REMOTES synchronized, even when a REMOTE changes a value locally the other REMOTES will receive a notification.

```javascript
var JsRO = new Thinfinity.JsRO({
   //'autoStart':false [optional]
});
```

- **Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoStart</td>
<td>(optional) allows you to set off automatically the linkage, by default is TRUE.</td>
</tr>
</tbody>
</table>

- **Returns**

  Thinfinity.JsRO Object.

- **Notes**

  To use this pseudo-class you must have got an existing HOST created.

- **Version**
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>Allows you to start linkage protocol manually with the HOST. It will be available when 'autoStart' is set to false.</td>
</tr>
<tr>
<td>on</td>
<td>Allows you to add an event to a specific node.</td>
</tr>
<tr>
<td>off</td>
<td>Allows you to remove an event from a specific node.</td>
</tr>
</tbody>
</table>

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>Model is just an entry point of the data object tree. All remote objects will be displayed as object-structure and can be accessed through their names.</td>
</tr>
</tbody>
</table>

9.5.2.1 Methods

9.5.2.1.1 start

```javascript
JsRO.start();
```

**Description**

Allows you to start linkage protocol manually with the HOST. It will be available when 'autoStart' is set to false.

**File**

Thinfinity.JsRO

9.5.2.1.2 on

```javascript
// The events can be:
// local (client side)
JsRO.on('model:Form1.checkBox1','changed',function(elem){  
  // yourCode
});
// remote (events declared on the server side)
JsRO.on('Form1.checkBox1','click', function(elem){
  // yourCode
});
```
### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>path of the node as string.</td>
</tr>
<tr>
<td>event</td>
<td>name of the event as string.</td>
</tr>
<tr>
<td>handler</td>
<td>callback as function.</td>
</tr>
</tbody>
</table>

### Returns

Returns boolean. TRUE if the event was added and FALSE when it was not.

### Description

Allows you to add an event to a specific node.

### File

**Thinfinity.JsRO**

#### 9.5.2.1.3 off

```javascript
// The events can be:
// local (client side)
JsRO.off('model:Form1.checkBox1', 'changed', handler);
// remote (events declared on the server side)
JsRO.off('Form1.checkBox1', 'click', handler);
```

### Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>node path as string.</td>
</tr>
<tr>
<td>event</td>
<td>event name as string.</td>
</tr>
<tr>
<td>handler</td>
<td>reference to the handler function previously set in the on method.</td>
</tr>
</tbody>
</table>

### Returns

Returns boolean. TRUE if the event was removed and FALSE when it was not.

### Description

Allows you to remove an event from a specific node.

### File

**Thinfinity.JsRO**
9.5.2.2 Properties

9.5.2.2.1 model

JavaScript

JsRO.model.yourObjectName.[n]

≡ Description

Model is just an entry point of the data object tree. All remote objects will be displayed as object-structure and can be accessed through their names.

≡ File

Thinfinity.JsRO