Table of Contents

Introduction .................................................. 21
Architecture .................................................................. 22
Getting Started .......................................................... 24
  1 Installing Thinfinity VirtualUI................................. 24
  2 Simple UI Remoting............................................... 27
    Compiling and Testing a WinForms Application ........... 28
    Compiling and Testing a Delphi Application ................ 30
    Compiling and Testing a C++ Application .................. 31
    Registering the Application in Thinfinity VirtualUI Server ................................................................ 34
    Accessing the App from the Web ............................... 35
    Application Execution Behavior .............................. 37
    List of available demos for download ....................... 38
  3 Adapting the Application ........................................... 39
    Customizing the Web Page ....................................... 40
      Preparing the Web Page ....................................... 41
      Customizing Settings ......................................... 43
    Programming VirtualUI ........................................... 45
      Handling VirtualUI Events .................................... 46
      Using ClientSettings from Javascript ...................... 48
Advanced Programming ............................................. 49
  1 Javascript Remote Objects ....................................... 49
    The Development Lab ............................................ 50
    Life cycle of the jsRO Objects .................................. 51
    Creating jsRO Objects .......................................... 53
    Updating Properties .............................................. 55
    OnPropertyChange(), OnSet() y OnGet() Events ........... 56
    jsRO Remote Calls ............................................... 60
      Remote Methods ............................................... 61
      Custom Events .................................................. 64
    Available demos for Javascript Remote Objects .......... 67
  2 Adding Web Content ............................................... 67
    The HTMLDoc Object ............................................ 68
      Loading Scripts .................................................. 69
      Importing HTML files ......................................... 70
      Accessing local files .......................................... 71
      Creating Web Component instances ....................... 72
    Web Components ............................................... 73
      Loading Web Component Scripts ............................ 74
      Importing Web Component HTML files ..................... 75
      Accessing local resources from the Web Component .... 76
Advanced Features .................................................. 77
  1 Scaling and Load Balancing ..................................... 77
    Scaling and Load Balancing Configurations ................ 77
    Installing components .......................................... 80
    Enabling Multiple RDS Accounts .............................. 81
Configuring Load Balancing ........................................................................................................ 83

2 Enhanced Browser and DPI Support ....................................................................................... 86
   Model Inheritance .................................................................................................................... 87
   Property Reference .................................................................................................................. 88
   The Calculation Process .......................................................................................................... 90
   Examples .................................................................................................................................. 91

3 One-Time URL .......................................................................................................................... 93
   How it Works ............................................................................................................................. 94
   Creating a One-Time URL ......................................................................................................... 96

4 Passing Command Line Arguments to VirtualUI Apps ............................................................ 98
   Setting Command Line Arguments in the Application Profile ................................................. 99
   Sending Command Line Arguments in the VirtualUI URL ...................................................... 102
   Building the URL in Javascript ............................................................................................... 104
   Combining Application Profile and URL Command Line Arguments ................................... 105

5 Recording and Playing Sessions ............................................................................................... 106
   Recording a Session ................................................................................................................ 107
      The OnRecorderChanged Event ........................................................................................ 108
   Playing Recorded Sessions ..................................................................................................... 110
   Creating your Own Player ........................................................................................................ 111

6 Recover Browser Information with BrowserInfo ...................................................................... 113

7 Change Browser Behavior Using ClientSettings ...................................................................... 116

8 End-User Authentication ........................................................................................................... 119
   Entering Credentials ............................................................................................................... 120
   Processing End-user Credentials ............................................................................................ 122
   Authentication Methods ......................................................................................................... 123
      Windows Logon .................................................................................................................... 123
      RADIUS .............................................................................................................................. 125
      OAuth 2.0 ............................................................................................................................. 128
      OAuth2Models.ini ............................................................................................................... 130
      CSS for SSO Options ........................................................................................................... 131
      External DLL ......................................................................................................................... 132
      Authentication API .............................................................................................................. 134

9 Virtualization ........................................................................................................................... 137
   File System Virtualization ....................................................................................................... 137
      Creating File System Redirections ......................................................................................... 138
      The .dir Files ......................................................................................................................... 142
      Testing the File System Virtualization .................................................................................. 143
   Registry Virtualization ............................................................................................................ 143
      Creating Registry Redirections ............................................................................................. 144
      The .reg Files ....................................................................................................................... 147
      Testing Registry Virtualization ............................................................................................ 148
   Implementing Virtualization in Your Application .................................................................... 148

10 OEM Licensing ......................................................................................................................... 149
   How to Create and Revoke Licenses ......................................................................................... 151
      Creating Licenses .................................................................................................................. 152
      Revoking Licenses .................................................................................................................. 155

11 Silent Install Options ............................................................................................................... 157

Web Interface ............................................................................................................................ 160
3 Gateway............................................................... 271

Appendix A - Dialogs ............................................... 274

1 Message Dialogs.................................................. 275
Message Dlg ................................................................ 276
Input Box .................................................................. 277
Formatted Message .................................................. 278

2 Printing Dialogs..................................................... 279
Print ....................................................................... 280
Page Setup .................................................................. 282
Print Preview ........................................................... 284

3 File Dialogs.......................................................... 285
Open File .................................................................. 286
Save As .................................................................... 288

Appendix B - Tailoring the interface ...................... 290

1 Customizing the Web Interface............................. 290
Changing the Logo .................................................. 291
Customizing the Web Files ...................................... 292
Files Location .......................................................... 293

Symbol Reference ................................................... 295

1 ActiveX Interfaces................................................ 295
Server Configuration Library .................................... 295

ILicense Interface ..................................................... 297

Contents
IProfile Interface .................................................................................................................. 304
Properties ................................................................................................................................. 305
  ID ............................................................................................................................................... 305
  Name ........................................................................................................................................ 306
  VirtualPath ............................................................................................................................... 306
  IsDefault ................................................................................................................................ 306
  Enabled ................................................................................................................................... 306
  ProfileKind ............................................................................................................................... 306
  FileName ................................................................................................................................ 307
  Arguments ................................................................................................................................. 307
  StartDir .................................................................................................................................. 307
  UserName ................................................................................................................................. 307
  Password ................................................................................................................................. 307
  ScreenResolution .................................................................................................................... 308
  WebLink .................................................................................................................................. 308
  HomePage ................................................................................................................................. 308
  IdleTimeout ............................................................................................................................... 308
  IconData .................................................................................................................................. 308
IProfile Properties ................................................................................................................... 308
  IProfile::ID Property .................................................................................................................. 309
  IProfile::Name Property ........................................................................................................... 309
  IProfile::VirtualPath Property ................................................................................................. 310
  IProfile::IsDefault Property .................................................................................................... 310
  IProfile::Enabled Property ....................................................................................................... 310
  IProfile::ProfileKind Property ............................................................................................... 310
  IProfile::FileName Property .................................................................................................... 311
  IProfile::Arguments Property .................................................................................................. 311
  IProfile::StartDir Property ...................................................................................................... 311
  IProfile::UserName Property ................................................................................................... 311
  IProfile::Password Property .................................................................................................... 312
  IProfile::ScreenResolution Property ...................................................................................... 312
  IProfile::WebLink Property ..................................................................................................... 312
  IProfile::HomePage Property ................................................................................................. 312
  IProfile::IdleTimeout Property ............................................................................................... 313
  IProfile::IconData Property .................................................................................................... 313
IProfiles Interface .................................................................................................................... 313
IProfiles Methods ..................................................................................................................... 314
  IProfiles::Add Method .............................................................................................................. 314
  IProfiles::Delete Method .......................................................................................................... 314
Properties .................................................................................................................................. 315
  Count ....................................................................................................................................... 315
  Item ......................................................................................................................................... 315
IBinding Interface ..................................................................................................................... 315
Properties .................................................................................................................................. 316
  Protocol .................................................................................................................................... 316
  IPAddress .................................................................................................................................. 316
  Port .......................................................................................................................................... 316
Binding Properties .................................................................................................................... 316
  IBinding::Protocol Property .................................................................................................... 316
  IBinding::IPAddress Property .................................................................................................. 317
  IBinding::Port Property ........................................................................................................... 317
ICertificate Interface ................................................................................................................ 317
Properties .................................................................................................................................. 318
  CertFile ................................................................................................................................... 318
CAFile .......................................................................................................................... 318
PKFile .......................................................................................................................... 318
PassPhrase ................................................................................................................... 318
ICertificate Properties ................................................................................................. 318
ICertificate::CertFile Property .................................................................................... 319
ICertificate::CAFile Property ..................................................................................... 319
ICertificate::PKFile Property ..................................................................................... 319
ICertificate::PassPhrase Property ................................................................................ 320
IRDS Interface ............................................................................................................ 320
Properties .................................................................................................................... 320
Enabled ........................................................................................................................ 320
UserName .................................................................................................................... 320
Password ...................................................................................................................... 321
IRDS Properties .......................................................................................................... 321
IRDS::UserName Property ............................................................................................ 321
IRDS::PassWord Property ........................................................................................... 321
IRDSAccounts Interface .............................................................................................. 322
IRDSAccounts Methods ............................................................................................ 322
IRDSAccounts::Item Method ....................................................................................... 322
IRDSAccounts::Add Method ....................................................................................... 323
IRDSAccounts::Delete Method ................................................................................... 323
IRDSAccounts Properties ............................................................................................ 323
IRDSAccounts::Count Property .................................................................................. 323
IGateway Interface ........................................................................................................ 324
IGateway Methods ....................................................................................................... 324
IGateway::Item Method ............................................................................................... 324
IGateway::Add Method ............................................................................................... 325
IGateway::Delete Method ........................................................................................... 325
IGateway Properties .................................................................................................... 325
IGateway::Count Property .......................................................................................... 326
Thinfinity Library .......................................................................................................... 326
Thinfinity::JSDataType Enumeration ............................................................................ 328
Thinfinity::Orientation Enumeration ............................................................................ 329
Thinfinity::MouseMoveGestureStyle Enumeration .................................................... 329
Thinfinity::MouseMoveGestureAction Enumeration .................................................. 329
Thinfinity::RecorderState Enumeration ..................................................................... 330
IVirtualUI Interface .................................................................................................... 330
IVirtualUI Methods ..................................................................................................... 331
IVirtualUI::Start Method .............................................................................................. 331
IVirtualUI::Stop Method ............................................................................................... 332
IVirtualUI::DownloadFile Method .............................................................................. 332
IVirtualUI::PrintPdf Method ....................................................................................... 333
IVirtualUI::OpenLinkDlg Method ................................................................................. 333
IVirtualUI::SendMessage Method ................................................................................ 333
IVirtualUI::AllowExecute Method .............................................................................. 334
IVirtualUI::SetImageQualityByWnd Method ................................................................ 334
IVirtualUI::UploadFile Method ................................................................................... 334
IVirtualUI::TakeScreenshot Method .......................................................................... 335
IVirtualUI::Preview Pdf Method .................................................................................. 335
IVirtualUI Properties ................................................................................................... 336
IVirtualUI::Active Property ......................................................................................... 336
IVirtualUI::Enabled Property ....................................................................................... 336
IVirtualUI::DevMode Property ..................................................................................... 337
IVirtualUI::StdDialogs Property .................................................................................. 337
IVirtualUI: BrowserInfo Property .......................................................... 337
IVirtualUI: DevServer Property ............................................................. 338
IVirtualUI: ClientSettings Property ...................................................... 338
IVirtualUI: Recorder Property ............................................................... 338
IVirtualUI: FileSystemFilter Property .................................................. 338
IVirtualUI: RegistryFilter Property ....................................................... 338
IVirtualUI: Options Property ............................................................... 339
IVirtualUI: HTMLDoc Property ............................................................. 339
IBrowserInfo Interface ........................................................................ 339
IBrowserInfo Methods ........................................................................ 340
IBrowserInfo::GetCookie Method ......................................................... 340
IBrowserInfo::SetCookie Method ......................................................... 340
IBrowserInfo Properties ....................................................................... 340
IBrowserInfo::ViewWidth Property ..................................................... 341
IBrowserInfo::ViewHeight Property ...................................................... 341
IBrowserInfo::BrowserWidth Property ................................................ 342
IBrowserInfo::BrowserHeight Property ............................................... 342
IBrowserInfo::ScreenWidth Property ................................................... 342
IBrowserInfo::ScreenHeight Property .................................................. 342
IBrowserInfo::Username Property ....................................................... 342
IBrowserInfo::IPAddress Property ....................................................... 343
IBrowserInfo::UserAgent Property ...................................................... 343
IBrowserInfo::ScreenResolution Property .......................................... 343
IBrowserInfo::Orientation Property .................................................... 343
IBrowserInfo::UniqueBrowserId Property ............................................ 344
IBrowserInfo::Location Property ......................................................... 344
IBrowserInfo::CustomData Property .................................................... 344
IBrowserInfo::SelectedRule Property .................................................. 344
IDevServer Interface ........................................................................... 344
IDevServer Properties .......................................................................... 345
IDevServer::Enabled Property ............................................................. 345
IDevServer::Port Property ................................................................... 345
IDevServer::StartBrowser Property ..................................................... 346
IClientSettings Interface ................................................................. 346
IClientSettings Properties ................................................................. 346
IClientSettings::MouseMoveGestureStyle Property ............................ 347
IClientSettings::MouseMoveGestureAction Property .......................... 347
IClientSettings::CursorVisible Property ............................................ 347
IJSValue Interface .............................................................................. 347
IJSValue Properties ............................................................................. 348
IJSValue::DataType Property .............................................................. 348
IJSValue::RawValue Property ............................................................... 348
IJSValue::AsString Property ............................................................... 349
IJSValue::AsInt Property ..................................................................... 349
IJSValue::AsBool Property ................................................................. 349
IJSValue::AsFloat Property ................................................................. 349
IJSValue::AsJSON Property ................................................................. 350
IJSNamedValue Interface ................................................................. 350
IJSNamedValue Properties ................................................................. 350
IJSNamedValue::Name Property .......................................................... 350
IJSObject Interface ............................................................................ 351
IJSObject Methods ............................................................................ 351
IJSObject::FireEvent Method .............................................................. 352
IJSObject::ApplyChanges Method ....................................................... 352
<table>
<thead>
<tr>
<th>IJSMethods Interface</th>
<th>367</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSMethods::Count Property</td>
<td>367</td>
</tr>
<tr>
<td>IJSMethods::Add Method</td>
<td>367</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IJSMethod Interface</th>
<th>366</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSMethod::ReturnValue Property</td>
<td>366</td>
</tr>
<tr>
<td>IJSMethod::Arguments Property</td>
<td>366</td>
</tr>
<tr>
<td>IJSMethod::Name Property</td>
<td>366</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IJSArguments Interface</th>
<th>365</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSArguments::Add Method</td>
<td>365</td>
</tr>
<tr>
<td>IJSArguments::Clear Method</td>
<td>365</td>
</tr>
<tr>
<td>IJSArguments::Item Method</td>
<td>365</td>
</tr>
<tr>
<td>IJSArguments::Remove Method</td>
<td>365</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IJSProperties Interface</th>
<th>364</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSProperties::Add Method</td>
<td>364</td>
</tr>
<tr>
<td>IJSProperties::Clear Method</td>
<td>364</td>
</tr>
<tr>
<td>IJSProperties::Remove Method</td>
<td>364</td>
</tr>
<tr>
<td>IJSProperties::ReadOnly Property</td>
<td>364</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IJSObjects Interface</th>
<th>363</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSObjects::Add Method</td>
<td>363</td>
</tr>
<tr>
<td>IJSObjects::Clear Method</td>
<td>363</td>
</tr>
<tr>
<td>IJSObjects::Item Method</td>
<td>363</td>
</tr>
<tr>
<td>IJSObjects::Remove Method</td>
<td>363</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IJSObject Interface</th>
<th>362</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSObject::Objects Property</td>
<td>362</td>
</tr>
<tr>
<td>IJSObject::Events Property</td>
<td>362</td>
</tr>
<tr>
<td>IJSObject::Methods Property</td>
<td>362</td>
</tr>
<tr>
<td>IJSObject::Properties Property</td>
<td>362</td>
</tr>
<tr>
<td>IJSObject::Id Property</td>
<td>362</td>
</tr>
</tbody>
</table>

| IJSObject::ApplyModel Method | 361 |
| IJSObject::Id Property | 361 |
| IJSObject::Properties Property | 361 |
| IJSObject::Methods Property | 361 |
| IJSObject::Events Property | 361 |
| IJSObject::Objects Property | 361 |

| IJSObject::ApplyModel Method | 360 |
| IJSObject::Id Property | 360 |
| IJSObject::Properties Property | 360 |
| IJSObject::Methods Property | 360 |
| IJSObject::Events Property | 360 |
| IJSObject::Objects Property | 360 |

| IJSObject::ApplyModel Method | 359 |
| IJSObject::Id Property | 359 |
| IJSObject::Properties Property | 359 |
| IJSObject::Methods Property | 359 |
| IJSObject::Events Property | 359 |
| IJSObject::Objects Property | 359 |

| IJSObject::ApplyModel Method | 358 |
| IJSObject::Id Property | 358 |
| IJSObject::Properties Property | 358 |
| IJSObject::Methods Property | 358 |
| IJSObject::Events Property | 358 |
| IJSObject::Objects Property | 358 |

| IJSObject::ApplyModel Method | 357 |
| IJSObject::Id Property | 357 |
| IJSObject::Properties Property | 357 |
| IJSObject::Methods Property | 357 |
| IJSObject::Events Property | 357 |
| IJSObject::Objects Property | 357 |

| IJSObject::ApplyModel Method | 356 |
| IJSObject::Id Property | 356 |
| IJSObject::Properties Property | 356 |
| IJSObject::Methods Property | 356 |
| IJSObject::Events Property | 356 |
| IJSObject::Objects Property | 356 |

| IJSObject::ApplyModel Method | 355 |
| IJSObject::Id Property | 355 |
| IJSObject::Properties Property | 355 |
| IJSObject::Methods Property | 355 |
| IJSObject::Events Property | 355 |
| IJSObject::Objects Property | 355 |

| IJSObject::ApplyModel Method | 354 |
| IJSObject::Id Property | 354 |
| IJSObject::Properties Property | 354 |
| IJSObject::Methods Property | 354 |
| IJSObject::Events Property | 354 |
| IJSObject::Objects Property | 354 |

<p>| IJSObject::ApplyModel Method | 353 |
| IJSObject::Id Property | 353 |
| IJSObject::Properties Property | 353 |
| IJSObject::Methods Property | 353 |
| IJSObject::Events Property | 353 |
| IJSObject::Objects Property | 353 |</p>
<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEvent</td>
<td>Event interface</td>
</tr>
<tr>
<td>IEvent Methods</td>
<td>Methods for event interface</td>
</tr>
<tr>
<td>IEvent::AddArgument Method</td>
<td>Method to add an event argument</td>
</tr>
<tr>
<td>IEvent::ArgumentAsNull Method</td>
<td>Method to cast an event argument to a null</td>
</tr>
<tr>
<td>IEvent::ArgumentAsString Method</td>
<td>Method to cast an event argument to a string</td>
</tr>
<tr>
<td>IEvent::ArgumentAsInt Method</td>
<td>Method to cast an event argument to an integer</td>
</tr>
<tr>
<td>IEvent::ArgumentAsFloat Method</td>
<td>Method to cast an event argument to a float</td>
</tr>
<tr>
<td>IEvent::ArgumentAsJSON Method</td>
<td>Method to cast an event argument to a JSON object</td>
</tr>
<tr>
<td>IEvent::Fire Method</td>
<td>Method to fire an event</td>
</tr>
<tr>
<td>IEvent Properties</td>
<td>Properties for event interface</td>
</tr>
<tr>
<td>IEvent::Name Property</td>
<td>Property for event name</td>
</tr>
<tr>
<td>IEvent::Arguments Property</td>
<td>Property for event arguments</td>
</tr>
<tr>
<td>IEvents</td>
<td>Event collection interface</td>
</tr>
<tr>
<td>IEvents Methods</td>
<td>Methods for event collection interface</td>
</tr>
<tr>
<td>IEvents::Item Method</td>
<td>Method to get an event by index</td>
</tr>
<tr>
<td>IEvents::Clear Method</td>
<td>Method to clear all events</td>
</tr>
<tr>
<td>IEvents::Add Method</td>
<td>Method to add an event</td>
</tr>
<tr>
<td>IEvents::Remove Method</td>
<td>Method to remove an event</td>
</tr>
<tr>
<td>IEvents Properties</td>
<td>Properties for event collection interface</td>
</tr>
<tr>
<td>IEvents::Count Property</td>
<td>Property for number of events</td>
</tr>
<tr>
<td>IJSBinding</td>
<td>Binding interface</td>
</tr>
<tr>
<td>IJSBinding Methods</td>
<td>Methods for binding interface</td>
</tr>
<tr>
<td>IJSBinding::Set Method</td>
<td>Method to set a binding variable</td>
</tr>
<tr>
<td>IJSCallback</td>
<td>Callback interface</td>
</tr>
<tr>
<td>IJSCallback Methods</td>
<td>Methods for callback interface</td>
</tr>
<tr>
<td>IJSCallback::Callback Method</td>
<td>Method to call a callback</td>
</tr>
<tr>
<td>IRecorder</td>
<td>Recorder interface</td>
</tr>
<tr>
<td>IRecorder Methods</td>
<td>Methods for recorder interface</td>
</tr>
<tr>
<td>IRecorder::Rec Method</td>
<td>Method to start recording</td>
</tr>
<tr>
<td>IRecorder::Pay Method</td>
<td>Method to play a recording</td>
</tr>
<tr>
<td>IRecorder::Stop Method</td>
<td>Method to stop recording</td>
</tr>
<tr>
<td>IRecorder Properties</td>
<td>Properties for recorder interface</td>
</tr>
<tr>
<td>IRecorder::Filename Property</td>
<td>Property for recording file name</td>
</tr>
<tr>
<td>IRecorder::Position Property</td>
<td>Property for recording position</td>
</tr>
<tr>
<td>IRecorder::Count Property</td>
<td>Property for number of recordings</td>
</tr>
<tr>
<td>IRecorder::State Property</td>
<td>Property for recording state</td>
</tr>
<tr>
<td>IRecorder::Options Property</td>
<td>Property for recording options</td>
</tr>
<tr>
<td>IRecorder::Tracks Property</td>
<td>Property for recording tracks</td>
</tr>
<tr>
<td>IRecTrack</td>
<td>Track interface</td>
</tr>
<tr>
<td>IRecTrack Properties</td>
<td>Properties for track interface</td>
</tr>
<tr>
<td>IRecTrack::Name Property</td>
<td>Property for track name</td>
</tr>
<tr>
<td>IRecTrack::Position Property</td>
<td>Property for track position</td>
</tr>
<tr>
<td>IRecTracks</td>
<td>Track collection interface</td>
</tr>
<tr>
<td>IRecTracks Methods</td>
<td>Methods for track collection interface</td>
</tr>
<tr>
<td>IRecTracks::Item Method</td>
<td>Method to get a track by index</td>
</tr>
<tr>
<td>IRecTracks Properties</td>
<td>Properties for track collection interface</td>
</tr>
<tr>
<td>IRecTracks::Count Property</td>
<td>Property for number of tracks</td>
</tr>
<tr>
<td>IFileSystemFilter</td>
<td>File system filter interface</td>
</tr>
<tr>
<td>IFileSystemFilter Methods</td>
<td>Methods for file system filter interface</td>
</tr>
<tr>
<td>IFileSystemFilter::User Property</td>
<td>Property for file system user</td>
</tr>
<tr>
<td>IFileSystemFilter::OrgFile Property</td>
<td>Property for file system organization file</td>
</tr>
<tr>
<td>IFileSystemFilter::Active Property</td>
<td>Property for file system active file</td>
</tr>
<tr>
<td>IRegistryFilter</td>
<td>Registry filter interface</td>
</tr>
<tr>
<td>IRegistryFilter Methods</td>
<td>Methods for registry filter interface</td>
</tr>
</tbody>
</table>

© 2018, Cybele Software, Inc.
Thinfinity® VirtualUI™
## Server Properties

- `Server.Binding Property` page 422
- `Server.Save Method` page 423
- `Server.Load Method` page 423

## Server Methods

- `Server.Server Constructor` page 424
- `Server.OnBrowserResize Event` page 424
- `Server.OnGetUploadDir Event` page 425
- `Server.OnClose Event` page 425
- `Server.OnReceiveMessage Event` page 425
- `Server.OnDownloadEnd Event` page 425
- `Server.OnRecorderChanged Event` page 426

## JSBinding Methods

- `JSBinding.JSBinding Constructor` page 427
- `JSBinding.Dispose Method` page 427
- `JSBinding.FireEvent Method` page 428
- `JSBinding.ApplyChanges Method` page 428
- `JSBinding.ApplyModel Method` page 428

## JSObject Methods

- `JSObject.OnPropertyChange Event` page 430
- `JSObject.Objects Property` page 430
- `JSObject.Methods Property` page 430
- `JSObject.Properties Property` page 430
- `JSObject.Id Property` page 430

## VirtualUI Events

- `VirtualUI.StdDialogs Property` page 421
- `VirtualUI.Options Property` page 422
- `VirtualUI.BrowserInfo Property` page 422
- `VirtualUI.HTMLDoc Property` page 422
- `VirtualUI.DevServer Property` page 422
- `VirtualUI.ClientSettings Property` page 423
- `VirtualUI.Recorder Property` page 423
- `VirtualUI.FileSystemFilter Property` page 423
- `VirtualUI.RegistryFilter Property` page 424

## VirtualUI Events

- `VirtualUI.OnBrowserResize Event` page 424
- `VirtualUI.OnGetUploadDir Event` page 425
- `VirtualUI.OnClose Event` page 425
- `VirtualUI.OnReceiveMessage Event` page 425
- `VirtualUI.OnDownloadEnd Event` page 425
- `VirtualUI.OnRecorderChanged Event` page 426

## JSObject Class

- `JSObject.JSObject Constructor` page 427
- `JSObject.JSObject Methods` page 427
- `JSObject.Dispose Method` page 427
- `JSObject.FireEvent Method` page 428
- `JSObject.ApplyChanges Method` page 428
- `JSObject.ApplyModel Method` page 428

## JSObject Properties

- `JSObject.Id Property` page 429
- `JSObject.Properties Property` page 429
- `JSObject.Methods Property` page 429
- `JSObject.Events Property` page 429
- `JSObject.Objects Property` page 430

## JSObject Events

- `JSObject.OnExecuteMethod Event` page 430
- `JSObject.OnPropertyChangeEvent` page 430

## JSBinding Class

- `JSBinding.JSBinding Constructor` page 431
- `JSBinding.JSBinding Methods` page 431
- `JSBinding.Dispose Method` page 432
- `JSBinding.Set Method` page 432

## JSCallback Class

- `JSCallback.JSCallback Constructor` page 433
- `JSCallback.JSCallback Methods` page 433
- `JSCallback.Dispose Method` page 433
- `JSCallback.Callback Method` page 433

## Cybele.Thinfinity.Settings.VirtualUI Namespace

- `Server.Class` page 434
- `Server.Server Constructor` page 435
- `Server.Server Methods` page 435
- `Server.Dispose Method` page 436
- `Server.Load Method` page 436
- `Server.Save Method` page 436
- `Server.HideSection Method` page 436
- `Server.Show Section Method` page 437

## Server Properties

- `Server.Binding Property` page 438
- `Server.Certificate Property` page 438
3 Delphi Classes......................................................................................................................................... 441

VirtualUI_SDK Namespace .................................................................................................................. 442

TBrowserInfo Class .............................................................................................................................. 442
  TBrowserInfo.Properties ..................................................................................................................... 443
    TBrowserInfo.ViewWidth Property ................................................................................................. 444
    TBrowserInfo.ViewHeight Property .............................................................................................. 444
    TBrowserInfo.BrowserWidth Property ......................................................................................... 444
    TBrowserInfo.BrowserHeight Property ......................................................................................... 444
    TBrowserInfo.ScreenWidth Property ............................................................................................ 444
    TBrowserInfo.ScreenHeight Property .......................................................................................... 444
    TBrowserInfo.Username Property .................................................................................................. 445
    TBrowserInfo.IPAddress Property ................................................................................................ 445
  TBrowserInfo.UserAgent Property .................................................................................................... 445
  TBrowserInfo.ScreenResolution Property ....................................................................................... 446
  TBrowserInfo.Orientation Property .................................................................................................. 446
  TBrowserInfo.UniqueBrowserId Property ......................................................................................... 446
  TBrowserInfo.Location Property ..................................................................................................... 446
  TBrowserInfo.CustomData Property ................................................................................................ 446
  TBrowserInfo.SelectedRule Property ............................................................................................... 447

TClientsSettings Class ........................................................................................................................ 447
  TClientsSettings.Properties ............................................................................................................... 447
    TClientsSettings.MouseMoveGestureStyle Property .................................................................... 448
    TClientsSettings.MouseMoveGestureAction Property .................................................................... 448
    TClientsSettings.CursorVisible Property ...................................................................................... 448

TDevServer Class ............................................................................................................................... 448
  TDevServer.CreateConstructor ......................................................................................................... 449
  TDevServer.Destroy Destructor ......................................................................................................... 449
  TDevServer.Properties ..................................................................................................................... 449
    TDevServer.Enabled Property .................................................................................................... 450
    TDevServer.Port Property ............................................................................................................. 450
    TDevServer.StartBrowser Property ............................................................................................. 450

TJSBinding Class ............................................................................................................................... 450

TJSCallback Class ............................................................................................................................... 451

TJSObject Class ................................................................................................................................... 451
  TJSObject.Methods ........................................................................................................................... 452
    TJSObject.FireEvent Method .......................................................................................................... 452
    TJSObject.ApplyChanges Method .................................................................................................. 453
    TJSObject.ApplyModel Method .................................................................................................... 453
  TJSObject.Properties ....................................................................................................................... 453
    TJSObject.Id Property .................................................................................................................... 454
    TJSObject.Properties Property ...................................................................................................... 454
    TJSObject.Methods Property ............................................................................................................ 454
    TJSObject.Events Property ............................................................................................................ 454
    TJSObject.Objects Property .......................................................................................................... 455
    TJSObject.Events Property ............................................................................................................ 455
VirtualUI_SDK.VirtualUI Function
VirtualUI_SDK.GetDllDir Function

THTMLDoc Class ................................................................. 455
THTMLDoc Methods.......................................................... 456
THTMLDoc.CreateSessionURL Method ................................ 456
THTMLDoc.createComponent Method ..................................... 457
THTMLDoc.GetSafeURL Method ........................................... 457
THTMLDoc.LoadScript Method ............................................. 458
THTMLDoc.ImportHTML Method .......................................... 458

TVirtualUI Class ............................................................... 458
TVirtualUI Methods................................................................ 459
    Dow nloadFile Method.................................................... 459
    TVirtualUI.Dow nloadFile Method (WideString, WideString, WideString) ............ 460
    TVirtualUI.Dow nloadFile Method (WideString, WideString) .................................. 460
    TVirtualUI.Dow nloadFile Method (WideString) ......................................................... 460
    Start Method .................................................................. 461
    TVirtualUI.Start Method (Integer) ...................................... 461
    TVirtualUI.Start Method () ................................................ 461
    UploadFile Method.......................................................... 462
    TVirtualUI.UploadFile Method (WideString) .......................... 462
    TVirtualUI.UploadFile Method () ......................................... 462
    TVirtualUI.Stop Method...................................................... 462
    TVirtualUI.PrintPdf Method................................................ 463
    TVirtualUI.PreviewPdf Method ............................................ 463
    TVirtualUI.TakeScreenshot Method ...................................... 463
    TVirtualUI.OpenLinkDlg Method ......................................... 464
    TVirtualUI.SendMessage Method ........................................ 464
    TVirtualUI.AllowExecute Method ......................................... 464
    TVirtualUI.SetImageQualityByWnd Method ............................ 465
    TVirtualUI.ShowVirtualKeyboard Method ............................ 465

TVirtualUI Properties.......................................................... 465
    TVirtualUI.Active Property ............................................... 466
    TVirtualUI.Enabled Property .............................................. 466
    TVirtualUI.DevMode Property ............................................. 467
    TVirtualUI.StdDialogs Property ......................................... 467
    TVirtualUI.BrowserInfo Property ........................................ 467
    TVirtualUI.DevServer Property .......................................... 467
    TVirtualUI.ClientSettings Property ...................................... 468
    TVirtualUI.Recorder Property ............................................ 468
    TVirtualUI.FileSystemFilter Property ................................... 469
    TVirtualUI.RegistryFilter Property .................................... 469
    TVirtualUI.Options Property ............................................. 469
    TVirtualUI.HTMLDoc Property ............................................ 469

TVirtualUI Events.................................................................. 469
    TVirtualUI.OnGetUploadDir Event .................................... 470
    TVirtualUI.OnBrowserResize Event .................................... 470
    TVirtualUI.OnReceiveMessage Event ................................. 470
    TVirtualUI.OnClose Event ................................................ 470
    TVirtualUI.OnDownloadEnd Event ..................................... 471
    TVirtualUI.OnRecorderChanged Event ............................... 471

VirtualUI_SDK.GetDllDir Function .......................................... 471
VirtualUI_SDK.VirtualUI Function ........................................... 472
VirtualUI_Settings Unit ....................................................... 472
VirtualUI_Settings.IID_JGateways Constant

VirtualUI_Settings.CLASS_JGateways Constant

VirtualUI_Settings.Gateways Type

IGateways Interface

IGateways Methods

IGateways.Get_Count Method

IGateways.Get_Item Method

IGateways.Add Method

IGateways.Delete Method

IGateways.Properties

IGateways.Count Property

IGateways.Item Property

IGatewaysDisp Interface

IGatewaysDisp Methods

IGatewaysDisp.Add Method

IGatewaysDisp.Delete Method

IGatewaysDisp.Properties

IGatewaysDisp.Count Property

IGatewaysDisp.Item Property

CoGateways Class

CoGateways Methods

CoGateways.Create Method

CoGateways.CreateRemote Method

TServer Class

TServer.Create Constructor

TServer.Destroy Destructor

TServer.Methods

TServer.Load Method

TServer.Save Method

TServer.HideSection Method

TServer.Show Section Method

TServer.Properties

TServer.Binding Property

TServer.RDSAccounts Property

TServer.Profiles Property

TServer.License Property

TServer.Gateways Property

VirtualUI_Settings.Base64ToIcon Function

VirtualUI_Settings.IconToBase64 Function

4 C++ Classes

CClientsClass

CClientsClass::CClientsClass Constructor

CClientsClass::CClientsClass Destructor

CClientsClass Methods

CursorVisible Method

CClientsClass::CursorVisible Method()

CClientsClass::CursorVisible Method(bool)

MouseMoveGestAction Method

CClientsClass::MouseMoveGestAction Method()

CClientsClass::MouseMoveGestAction Method(MouseMoveGestureAction)

MouseMoveGestStyle Method

CClientsClass::MouseMoveGestStyle Method()
CDevServer::Enabled Method () ........................................................................................................... 505
CDevServer::Enabled Method (bool) ........................................................................................................ 505
Port Method .................................................................................................................................................. 505
CDevServer::Port Method () .................................................................................................................... 506
CDevServer::Port Method (int) .................................................................................................................. 506
CDevServer::StartBrowser Method........................................................................................................... 506
CDevServer Properties .................................................................................................................................. 506
CDevServer::Enabled Property ................................................................................................................. 507
CDevServer::Port Property ....................................................................................................................... 507
CDevServer::StartBrowser Property ....................................................................................................... 507
JSObject Class ........................................................................................................................................... 508
JSObject::JSObject Constructor ................................................................................................................ 508
JSObject::JSObject Destructor .................................................................................................................. 508
JSObject Methods ...................................................................................................................................... 509
FireEvent Method ....................................................................................................................................... 509
JSObject::FireEvent Method (BSTR, IJSArguments *) .............................................................................. 510
JSObject::FireEvent Method (std::wstring, IJSArguments *) ................................................................. 510
Id Method ................................................................................................................................................... 510
JSObject::Id Method () ............................................................................................................................ 510
JSObject:QueryInterface Method ................................................................................................................ 511
JSObject::ApplyChanges Method ............................................................................................................... 511
JSObject::ApplyModel Method .................................................................................................................. 511
JSObject::Properties Method .................................................................................................................... 511
JSObject::Methods Method ....................................................................................................................... 512
JSObject::Events Method ........................................................................................................................... 512
JSObject::Objects Method ......................................................................................................................... 512
JSObject Properties .................................................................................................................................... 512
JSObject::Id Property .................................................................................................................................. 513
JSObject::Properties Property .................................................................................................................. 513
JSObject::Methods Property ..................................................................................................................... 513
JSObject::Events Property ....................................................................................................................... 513
JSObject::Objects Property ...................................................................................................................... 514
JSCallback Class ....................................................................................................................................... 514
JSCallback::JSCallback Constructor ........................................................................................................ 514
JSCallback Methods .................................................................................................................................... 515
JSCallback::Callback Method ..................................................................................................................... 515
JSBinding Class .......................................................................................................................................... 515
JSBinding::JSBinding Constructor ............................................................................................................ 516
JSBinding Methods ...................................................................................................................................... 516
JSBinding::Set Method .................................................................................................................................. 516
VirtualUI ..................................................................................................................................................... 516
CRecorder Class ......................................................................................................................................... 517
CRecorder::CRecorder Constructor ........................................................................................................ 517
CRecorder::CRecorder Destructor ........................................................................................................... 517
CRecorder Methods ..................................................................................................................................... 518
Filename Method .......................................................................................................................................... 518
CRecorder::Filename Method () .............................................................................................................. 519
CRecorder::Filename Method (std::wstring) ............................................................................................ 519
Options Method .......................................................................................................................................... 519
CRecorder::Options Method () ............................................................................................................... 519
CRecorder::Options Method (long) ........................................................................................................... 520
CRecorder::Position Method ...................................................................................................................... 520
CRecorder::Count Method ......................................................................................................................... 520
CRecorder::State Method ........................................................................................................................... 520
© 2018, Cybele Software, Inc.
<table>
<thead>
<tr>
<th>Class</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRecorder</td>
<td>521</td>
</tr>
<tr>
<td>CRecorder::Tracks Method</td>
<td>521</td>
</tr>
<tr>
<td>CRecorder::Rec Method</td>
<td>521</td>
</tr>
<tr>
<td>CRecorder::Play Method</td>
<td>521</td>
</tr>
<tr>
<td>CRecorder::Stop Method</td>
<td>522</td>
</tr>
<tr>
<td>CRecorder Properties</td>
<td>522</td>
</tr>
<tr>
<td>CRecorder::Tracks Property</td>
<td>522</td>
</tr>
<tr>
<td>CRecTracks Class</td>
<td>522</td>
</tr>
<tr>
<td>CRecTracks::CRecTracks Constructor</td>
<td>523</td>
</tr>
<tr>
<td>CRecTracks::~CRecTracks Destructor</td>
<td>523</td>
</tr>
<tr>
<td>CRecTracks Methods</td>
<td>523</td>
</tr>
<tr>
<td>CRecTracks::Item Method</td>
<td>524</td>
</tr>
<tr>
<td>CRecTracks::Count Method</td>
<td>524</td>
</tr>
<tr>
<td>CRecTracks Properties</td>
<td>524</td>
</tr>
<tr>
<td>CRecTracks::Item Property</td>
<td>524</td>
</tr>
<tr>
<td>CFFileSystemFilter Class</td>
<td>525</td>
</tr>
<tr>
<td>CFFileSystemFilter::CFFileSystemFilter Constructor</td>
<td>525</td>
</tr>
<tr>
<td>CFFileSystemFilter::~CFFileSystemFilter Destructor</td>
<td>525</td>
</tr>
<tr>
<td>CFFileSystemFilter Methods</td>
<td>526</td>
</tr>
<tr>
<td>Active Method</td>
<td>526</td>
</tr>
<tr>
<td>CFFileSystemFilter::Active Method ()</td>
<td>526</td>
</tr>
<tr>
<td>CFFileSystemFilter::Active Method (bool)</td>
<td>526</td>
</tr>
<tr>
<td>CfgFile Method</td>
<td>527</td>
</tr>
<tr>
<td>CFFileSystemFilter::CfgFile Method ()</td>
<td>527</td>
</tr>
<tr>
<td>CFFileSystemFilter::CfgFile Method (std::wstring)</td>
<td>527</td>
</tr>
<tr>
<td>User Method</td>
<td>527</td>
</tr>
<tr>
<td>CFFileSystemFilter::User Method ()</td>
<td>528</td>
</tr>
<tr>
<td>CFFileSystemFilter::User Method (std::wstring)</td>
<td>528</td>
</tr>
<tr>
<td>CFFileSystemFilter Properties</td>
<td>528</td>
</tr>
<tr>
<td>CFFileSystemFilter::User Property</td>
<td>528</td>
</tr>
<tr>
<td>CFFileSystemFilter::CfgFile Property</td>
<td>529</td>
</tr>
<tr>
<td>CFFileSystemFilter::Active Property</td>
<td>529</td>
</tr>
<tr>
<td>CRegistryFilter Class</td>
<td>529</td>
</tr>
<tr>
<td>CRegistryFilter::CRegistryFilter Constructor</td>
<td>530</td>
</tr>
<tr>
<td>CRegistryFilter::~CRegistryFilter Destructor</td>
<td>530</td>
</tr>
<tr>
<td>CRegistryFilter Methods</td>
<td>530</td>
</tr>
<tr>
<td>Active Method</td>
<td>531</td>
</tr>
<tr>
<td>CRegistryFilter::Active Method ()</td>
<td>531</td>
</tr>
<tr>
<td>CRegistryFilter::Active Method (bool)</td>
<td>531</td>
</tr>
<tr>
<td>CfgFile Method</td>
<td>531</td>
</tr>
<tr>
<td>CRegistryFilter::CfgFile Method ()</td>
<td>532</td>
</tr>
<tr>
<td>CRegistryFilter::CfgFile Method (std::wstring)</td>
<td>532</td>
</tr>
<tr>
<td>User Method</td>
<td>532</td>
</tr>
<tr>
<td>CRegistryFilter::User Method ()</td>
<td>533</td>
</tr>
<tr>
<td>CRegistryFilter::User Method (std::wstring)</td>
<td>533</td>
</tr>
<tr>
<td>CRegistryFilter Properties</td>
<td>533</td>
</tr>
<tr>
<td>CRegistryFilter::User Property</td>
<td>533</td>
</tr>
<tr>
<td>CRegistryFilter::CfgFile Property</td>
<td>534</td>
</tr>
<tr>
<td>CRegistryFilter::Active Property</td>
<td>534</td>
</tr>
<tr>
<td>CHTMLDoc Class</td>
<td>534</td>
</tr>
<tr>
<td>CHTMLDoc::CHTMLDoc Constructor</td>
<td>534</td>
</tr>
<tr>
<td>CHTMLDoc::~CHTMLDoc Destructor</td>
<td>535</td>
</tr>
<tr>
<td>CHTMLDoc Methods</td>
<td>535</td>
</tr>
<tr>
<td>CHTMLDoc::CreateSessionURL Method</td>
<td>536</td>
</tr>
<tr>
<td>CHTMLDoc::CreateComponent Method</td>
<td>536</td>
</tr>
</tbody>
</table>
5 VirtualUI.sdk.min.js

Thinfinity.VirtualUI

Events
- onLoading
- onShow
- onClose
- onError

Properties
- devMode

Thinfinity.JsRO

Methods
- start
- on
- off

Properties
- model
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 Certificate

Copyright © 2018, 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 10
  - Microsoft Windows Server 2012
  - Microsoft Windows Server 2016

- A development environment:
  - Microsoft Visual Studio
  - Delphi 5 to 10.1 Berlin
  - C++ Builder
  - Microsoft Visual Basic (including VB 6.0)
  - 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 10
- Windows Server 2012 and 2012 R2
- Windows Server 2016

**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, Edge, 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
  - 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 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)

**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.

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
- Application Execution Behavior
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 in debug mode.

You will see this message:
If you press the 'Start Web Browser' button, a web browser will open pointing to the application. The application will also open in desktop mode.

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.

```pascal
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 in debug mode.

You will see this message:

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`.

Start Web Browser

[ ] Do not show again
If you press the 'Start Web Browser' button, a web browser will open pointing to the application. The application will also open in desktop mode.

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:

```
#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;
}
8. Compile the program and run it in debug mode.

You will see this message:

![Welcome to Thinfinity VirtualUI]

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.

If you press the 'Start Web Browser' button, a web browser will open pointing to the application. The application will also open in desktop mode.

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.
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

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.

3. Authenticated users can log out with the 'Logout' button in the top right corner of the web interface.

   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.

3. Authenticated users can log out with the 'Logout' button in the top right corner of the web interface.
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.
3.2.7 List of available demos for download

Below you'll find a list for all the available demos for Thinfinity VirtualUI:

- Microsoft Access:
  https://www.cybelesoft.com/support/files/DemosVUI/VirtualUI_Access.zip

- Windows Presentation Foundation:
  https://www.cybelesoft.com/support/files/DemosVUI/WpfApplication1.zip

- Visual Fox Pro:
  https://www.cybelesoft.com/support/files/DemosVUI/VisualFoxPro.zip

- Visual Fox Pro 9:

- Visual Basic 6:
  https://www.cybelesoft.com/support/files/DemosVUI/VB6_Upload.zip

- Visual Basic .Net:
  https://www.cybelesoft.com/support/files/DemosVUI/VBNet.zip

- Delphi:
  https://www.cybelesoft.com/support/files/DemosVUI/UploadDemoOnUploadEnd.zip
  https://www.cybelesoft.com/support/files/DemosVUI/VCLFileTest.zip
  https://www.cybelesoft.com/support/files/DemosVUI/Delphi_ShellExecute.zip
  https://www.cybelesoft.com/support/files/DemosVUI/LibConfigCreateProfile.zip

- Lazarus:
  https://www.cybelesoft.com/support/files/DemosVUI/Lazarus.zip

- C++:
  https://www.cybelesoft.com/support/files/DemosVUI/CppBuilder.zip

- C#:
  https://www.cybelesoft.com/support/files/DemosVUI/cSharpDemo.zip
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密度Dpi=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'.

![Thinfinity VirtualUI Application Profiles Editor](image-url)
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 the following classes:

- The Thinfinity.VirtualUI class
- The Thinfinity.JsRO class

**Thinfinitiy.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
3.3.2.2 Using ClientSettings from Javascript

ClientSettings is an additional interface available in the Thinfinity VirtualUI Library that allows developers to programmatically configure some browser environment layout and behavior settings from their applications (e.g., the mouse movement type). This interface is available both in the application (C++, C#, Delphi, etc.) and in the browser side, where it was extended to manage some web interface elements (the DockMenu widget, for the moment). ClientSettings is available in Javascript as a Thinfinity.VirtualUI attribute:

```javascript
var vui = new Thinfinity.VirtualUI();
var clientSettings = vui.ClientSettings;
```

All ClientSettings attributes have a default value that determines the typical behavior in each case. Developers can change it by setting new values to these attributes, which override the defaults before the connection starts. To hide the mouse pointer and remove the “Error Reporting” icon from the DockMenu, just do the following:

```javascript
var vui = new Thinfinity.VirtualUI();
var clientSettings = vui.ClientSettings;

clientSettings.CursorVisible = false;
clientSettings.DockMenu.ErrorReporting = false;
```

ClientSettings Applying Order

As most of the ClientSettings values can be applied both in the browser side and the application, we need to define an applying order:

- A value set in the application overrides the default value.
- Any value set in the browser side always overrides any other value.

Read more about how to change browser behavior using ClientSettings.
4 Advanced Programming

Thinfinity VirtualUI offers a rich API that allows for a tight integration with external web resources.

**JavaScript Remote Objects (jsRO)** makes possible to define remote objects and expose properties, methods and events that are synchronized in both ends, the application and the web browser.

**Web Components** specialized methods let you embed HTML and Javascript components that can enhance your application with web browser content.

**Read more:**
- [JavaScript Remote Objects (jsRO)]
- [Web Components]

4.1 Javascript Remote Objects

JavaScript Remote Objects (**jsRO**) 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]
- [OnPropertyChanged(), OnSet() and OnGet() Events]
- [jsRO Remote Calls]
4.1.1 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](#)
- [OnPropertyChanged(), OnSet() and OnGet() Events](#)
- [jsRO Remote Calls](#)
4.1.2 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:

```javascript
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:

```javascript
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:

```javascript
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.

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

This last event will also be triggered when closing the application.
Read more:
- Creating jsRO Objects
- Updating Properties
- OnPropertyChange(), OnSet() and OnGet() Events
- jsRO Remote Calls
4.1.3 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:
Read more:
- Updating Properties
- OnPropertyChange(), OnSet() and OnGet() Events
- jsRO Remote Calls
4.1.4 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()` and `OnGet()` Events
- JsRO Remote Calls
4.1.5 OnPropertyChanged(), 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

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:

```delphi
// 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
```

© 2018, Cybele Software, Inc.
```javascript
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;
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("^[\0-9,a-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:

```javascript
ro.backgroundColor = "#FF0000";
```
4.1.6 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
4.1.6.1 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:

```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", JSDT_FLOAT) // 1st number to multiply
  .AddArgument("b", JSDT_FLOAT) // 2nd number to multiply
  .OnCall(new JSCallback( // Adds the callback
delegate(IJSObject parent, IJSMethod Method)
{
  float a, b;
  a = Method.Arguments["a"].AsFloat;
  b = Method.Arguments["b"].AsFloat;
  Method.ReturnValue.AsFloat = a * b;
}).ReturnValue.DataType = JSDT_FLOAT;
ro.ApplyModel();
```

Method definition in C#:

```
// Creates the remote object
ro = new JSObject("ro");
// Adds the method
ro.Methods.Add("multiply") // Returns a JSMethod
  .AddArgument("a", IJSDataType.JSDT_FLOAT) // 1st number to multiply
  .AddArgument("b", IJSDataType.JSDT_FLOAT) // 2nd number to multiply
  .OnCall(new JSCallback( // Adds the callback
delegate(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;
```
b = Method.Arguments["b"].AsFloat;
Method.ReturnValue.AsFloat = a * b;
});
ro.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
4.1.6.2 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
```

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:
FRo.Events['mousePositionChanged']
  .ArgumentAsJSON('coords', '{"x": ' + X + ', "y": ' + Y + '}')
  .Fire;
end;

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();
}
```

Definition and use of an event in C#:

```java
// Creates the remote object
ro = new JSObject("ro");
// Adds the event
ro.Events.Add("mousePositionChanged")
  .AddArgument("coords", IJSDataType.JSDT_JSON) // 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) {
```
```javascript
console.log('mouse moved to [' + coords.x + ', ' + coords.y + ']');
```
4.1.7 Available demos for Javascript Remote Objects

Below you'll find the available demos for JSRO and Thinfinity VirtualUI:

- **Vui-Video** = This demo uses JSRO to transmit a video for rendering in the browser.
  

- **Vui-Audio** = This demo uses JSRO to send audio directly to the browser.


- **Vui-QZ** = This demo uses JSRO to scan barcodes and send them to the application.


- **Vui-Webcam** = This demo uses JSRO to access the user Webcam and transmit it to the application.

  .https://www.cybelesoft.com/support/files/DemosVUI/vui-webcam.7z

4.2 Adding Web Content

We took VirtualUI integration tools one step further and created the HTMLDoc Object to enhance the integration of Windows app resources to the web.

This object handles web elements from the server side, sparing the developer the need to learn javascript to include interactive web content in their application. This makes the web enabling process much more straightforward in terms of time and organization and empowers developers to easily enhance their VirtualUI web enabled application with reusable widgets and components in web documents and web applications.

This API allows your to take advantage of Web Components and can also be used to include and run Javascript files that do not follow the HTML5 Web Component specification.

Read more:
- The HTMLDoc Object
- Web Components
4.2.1 The HTMLDoc Object

VirtualUI provides a specialized application-side object called HTMLDoc, with methods that allow you to add web content without changing the original HTML or javascript files.

- **Load and run Javascript files** from the Windows app.
- **Import HTML files** from the Windows app.
- **Securely access files** not located in the web tree (needed to deploy web or media files along with your application).
- **Create an HTML5 Web Component** defined in an imported HTML file.
4.2.1.1 Loading Scripts

Thinfinity VirtualUI allows you to load and run Javascript files from the Windows app. In order to load a new javascript file, you can use the LoadScript method of the HTMLDoc component:

```
VirtualUI.HTMLDoc.LoadScript(URL, [Filename])
```

- The "URL" parameter is the Uniform Resource Locator pointing to the Javascript file location in the net.
- "Filename" is an optional parameter that, when specified, points to the actual Javascript file in the local file system. In this case the "URL" parameter will be virtual, and will resolve to this local "filename".
4.2.1.2 Importing HTML files

Thinfinity VirtualUI allows you to insert HTML content into the current web page. You can do this from your Windows app by calling the `ImportHTML` method of the `HTMLDoc` component:

```
VirtualUI.HTMLDoc.ImportHTML(URL, [Filename])
```

- The "URL" parameter is the Uniform Resource Locator pointing to the HTML file location in the net.
- "Filename" is an optional parameter that, when specified, points to the actual HTML file in the local file system. In this case, the "URL" parameter will be virtual, and will translate to this local "filename".
4.2.1.3 Accessing local files

With the purpose of providing secure urls to files that are not located in the application's web tree, we included two methods:

**VirtualUI.HTMLDoc.CreateSessionURL(URL, LocalPath)**

This method allows you to create a virtual URL valid during the session's lifetime.

- The "URL" parameter is the virtual URL to access the Filename or Directory specified in LocalPath.
- "LocalPath" can be a file name or a directory. If it is a file name, the URL parameter resolves to that filename; if it is a directory, the URL parameter is a virtual path that can be used to access any file under this local directory.

**URL = VirtualUI.HTMLDoc.GetUniqueURL(Filename, lifetime)**

This method allows you to create a unique virtual URL for a specific file, valid during the time specified in minutes in the Lifetime parameter.

- The "Filename" parameter points to the local filename we want to create an unique URL for.
- "Lifetime" is the time during which the returned URL will be valid.
4.2.1.4 Creating Web Component instances

To create a Web Component instance, we added a specific method that will allow you to embed the component in the web page, tied or untied to a Windows control.

When the Web Component is tied to a Windows control, it will look like it is "embedded" in the form, replacing the Windows control area, following its position and size when it changes. When untied, it will be embedded into the web page, but its position and size will not be attached to any control area.

VirtualUI.HTMLDoc.CreateComponent(Id, TagOrHTML, ReplaceWnd)

- "Id" parameter is the HTML element ID.
- "TagOrHTML" can be a tagname or a valid HTML string.
- "ReplaceWnd" is the handle to the window control to be "replaced".

Examples:

1. Adding a tied Web Component by the tagname:

VirtualUI.HTMLDoc.CreateComponent("video1", "vui-video", panel1.Handle)

In this example, the HTML "<vui-video></vui-video>" will be inserted, and dynamically positioned according to panel1's position and size.

2. Adding tied HTML content from a valid HTML string:

VirtualUI.HTMLDoc.CreateComponent("hworld", "<div><span>hello world</span></div>", panel1.Handle)

In this example, the HTML "<div><span>hello world</span></div>" will be inserted, and dynamically positioned according to panel1's position and size.

3. Adding an untied Web Component by the tagname:

VirtualUI.HTMLDoc.CreateComponent("qzprinter1", "qzprinter")

In this example, the Web Component will be inserted, but not positioned according to any visible control.
4.2.2 Web Components

Web Components are a new and exciting technology that takes the well known "reusable software components" concept to the web. With this HTML5 specification you can build HTML+javascript pieces and use them as a custom HTML element.

In Thinfinity VirtualUI you can easily embed Web Components in your Windows application. Some Web Components examples are: a native video streaming component, a raw printing component, a QR component, a signature component, a Google Maps component, and so on.

With the purpose of embedding a Web Component in a Windows application, we provide a specialized object that allows you to:

- Load and run Web Component Javascript files from the Windows app.
- Import Web Component HTML files from the Windows app
- Securely access local resources that are not in the application's web tree (needed to deploy web or media files in your application)
4.2.2.1 Loading Web Component Scripts

Let's suppose that you are about to include some new functionality and logic which may need some Javascript library that is currently not loaded. In order to load this new javascript library you can use the `LoadScript` method of the HTMLDoc component.
4.2.2.2 Importing Web Component HTML files

Web Components are defined in HTML files. In order to be able to create a Web Component, you need to import that file into the HTML DOM. With VirtualUI you can do this from your Windows app by calling the `ImportHTML` method of the `HTMLDoc` component.
4.2.2.3 Accessing local resources from the Web Component

Let's suppose we include a web component that needs to display a signature, and the signature image file is somewhere in the file system. Complying with standard web server specifications, Thinfinity VirtualUI forbids the access to files that are not in the root web tree. This means that to access that image file, you would be forced to first place it in the application's web tree. This wouldn't be neat!

With the purpose of providing secure urls to files that are not within the web tree root, we included the CreateSessionURL and GetUniqueURL methods. These methods help you create virtual URLs for the application session's lifetime, or for a specific file and valid only during a specified lifetime, in minutes.
5 Advanced Features

Take a look at the Thinfinity VirtualUI advanced features:

- Enhanced Browser and DPI Support
- End-User Authentication
- One-Time URL
- Passing Command Line Arguments to VirtualUI Apps
- Recording and Playing Sessions
- Scaling and Load Balancing
- Printing
- Virtualization
- OEM Licensing

5.1 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

5.1.1 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**

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
5.1.2 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

5.1.3 Enabling Multiple RDS Accounts

In order to enable multiple RDS accounts, go to the 'RDS' tab and check the 'Run under this account' checkbox:

Thinfinit® VirtualUI™ requires at least one interactive Windows session. By default it uses the console session, sharing this session among all connected users.

You can configure Thinfinit® VirtualUI™ to run under an alternate Windows session or, if you installed the Gateway, you can choose to balance memory usage/performance by configuring one session per user or distribute users evenly among a number of Windows sessions.

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.

<table>
<thead>
<tr>
<th>Session sharing</th>
<th>Checking this option will split all new application instances in the amount of RDS users established in the &quot;Session count&quot; field.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session count</td>
<td>Establishes the maximum amount of RDS users that will spawn on the server.</td>
</tr>
</tbody>
</table>

Read more:

- [Scaling and Load Balancing](#)
### 5.1.4 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.

---

**Gateway Manager**

- **File**
- **Help**
- **General**
  - Internal Broker Server

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

- **Network ID**

  Server started. Listening http on port 6580.

#### 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
5.2 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:

```
[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 settings to apply, the connection’s ‘Resolution’ property must be set to ‘Fit to browser’, and the BrowserRules.ini file must be pointed in the application profile.

Configure this setting in for your application in the 'General' tab of the Application Profile, or the 'General' tab of the Application Profile in the development server.

Read more:
- Model Inheritance
- Property Reference
- The Calculation Process
- Examples

5.2.1 Model Inheritance

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.

<p>| match-device-pixel-ratio | Matches any device with a specific pixel ratio. |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>match-mobile</td>
<td>Matches any mobile device.</td>
</tr>
<tr>
<td>match-orientation</td>
<td>Matches any device with the specified orientation: landscape or portrait.</td>
</tr>
<tr>
<td>match-screen-height-range</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>match-screen-width-range</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>match-screen-height</td>
<td>Matches any device with a specified screen height.</td>
</tr>
<tr>
<td>match-screen-width</td>
<td>Matches any device with a specified screen width.</td>
</tr>
<tr>
<td>match-user-agent</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>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent-model</td>
<td>Establish the parent model for this model.</td>
</tr>
</tbody>
</table>

The following properties deal with the display resolution:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>device-pixel-ratio</td>
<td>Overrides the original device pixel ratio, scaling the content accordingly.</td>
</tr>
<tr>
<td>max-device-pixel-ratio</td>
<td>This property determines the maximum device pixel ratio accepted. The lesser of the device’s device pixel ratio and this value is applied to scale the display.</td>
</tr>
</tbody>
</table>

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>
</tbody>
</table>
max-height | Remote desktop maximum height.
max-width | Remote desktop maximum width.
min-height | Remote desktop minimum height.
min-width | Remote desktop minimum width.

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.

margin-left | Width of an area at the left of the device screen that will not be used for displaying the remote desktop.
margin-bottom | Width of an area at the bottom of the device screen that will not be used for displaying the connection.
margin-right | Width of an area at the right of the device screen that will not be used for displaying the connection.
margin-top | Width of an area at the top of the device screen that will not be used for displaying the connection.

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

5.2.3 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](#)

### 5.2.4 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

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

[ipad]
parent-model = mobile
match-user-agent = ipad

[iphone4]
parent-model = mobile
match-user-agent = iphone
match-screen-width = 480
match-screen-height = 320
device-pixel-ratio = 1.5
```

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>min-width</td>
<td>640</td>
</tr>
<tr>
<td>min-height</td>
<td>480</td>
</tr>
<tr>
<td>max-width</td>
<td>2560</td>
</tr>
<tr>
<td>max-height</td>
<td>1600</td>
</tr>
<tr>
<td>max-device-pixel-ratio</td>
<td>2</td>
</tr>
</tbody>
</table>

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 ipad 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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>min-width</td>
<td>640</td>
</tr>
<tr>
<td>min-height</td>
<td>480</td>
</tr>
<tr>
<td>max-width</td>
<td>2560</td>
</tr>
<tr>
<td>max-height</td>
<td>1600</td>
</tr>
<tr>
<td>max-device-pixel-ratio</td>
<td>2</td>
</tr>
<tr>
<td>device-pixel-ratio</td>
<td>1.5</td>
</tr>
</tbody>
</table>
5.3 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:

- 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](#)
5.3.1 How it Works

The One-Time URL is a 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.

A usual scenario involves a backend service (ie. a web server), where the user’s credentials are validated. This backend service communicates with a VirtualUI 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
5.3.2 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>
<tr>
<td>customdata</td>
<td>optional</td>
<td>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.</td>
</tr>
<tr>
<td>plen</td>
<td>optional</td>
<td>Length of the passcode to be returned.</td>
</tr>
<tr>
<td>expires</td>
<td>optional</td>
<td>Ticket expiration time, in minutes.</td>
</tr>
</tbody>
</table>

**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 a 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
5.4 Passing Command Line Arguments to VirtualUI Apps

An application can accept command line arguments when invoked. This allows the user to send additional information to the application when it is executed. When an application is launched, the OS passes the command line arguments to the application as a collection of string values, using a white-space as separator.

Applications receive external arguments on a regular basis: when you open a document by clicking on its icon, the OS selects the associated executable program and calls it, sending the full document filename to the program as an argument.

Thinfinity VirtualUI allows you to send external arguments to applications in a transparent way, which works exactly like sending arguments from the command line.

Read more:
- Setting Command Line Arguments in the Application Profile
- Sending Command Line Arguments in the VirtualUI URL
- Building the URL in Javascript
- Combining Application Profile and URL Command Line Arguments
5.4.1 Setting Command Line Arguments in the Application Profile

Thinfinity VirtualUI enables external argument definition from the VirtualUI Server Manager. When you create an application profile, you can specify the execution arguments in the General tab panel of the Application Profiles editor.

These arguments will be received by the application as a list of string values, using the white-space character as argument delimiter.
In Delphi you can get command line arguments using the System.ParamCount and System.ParamStr methods. ParamCount returns the number of arguments, and ParamStr(index) returns each argument, with ParamStr(0) always being the path and name of the invoked program. Here is a way you can show the received arguments list:

```
Writeln(Format('Arguments = %d', [ParamCount]))
for p := 1 to ParamCount do
  Writeln(Format('argument %d = "%s"', [p, ParamStr(p)]));
```

In C++, both the argument counter and the argument list will be received in the two main() function arguments. Like in Delphi, the first argument in the string array is the path and name of the program itself:

```
void main(int argCounter, char* arguments[]) {
  cout << "Arguments = " << argCounter << endl;
  for(int i = 1; i < argCounter; i++)
    cout << "arguments[" << i << "] = " << arguments[i] << endl;
}
```

You can also write the main function header in this way:
Unlike Delphi and C++, C# excludes the name and path of the program from the arguments collection. Also, C# offers two ways to get the arguments, by using a traditional for:

```csharp
public static void Main(string[] arguments)
{
    Console.WriteLine("Arguments = \{0\}", args.Length);
    for(int i = 0; i < arguments.Length; i++)
    {
        Console.WriteLine("Arguments[{0}] = \{1\}", i, arguments[i]);
    } 
}
```

or by using the foreach:

```csharp
foreach(string s in arguments)
{
    Console.WriteLine(s);
}
```

**Read more:**
- [Sending Command Line Arguments in the VirtualUI URL](#)
- [Building the URL in Javascript](#)
- [Combining Application Profile and URL Command Line Arguments](#)
5.4.2 Sending Command Line Arguments in the VirtualUI URL

VirtualUI allows you to send external arguments in the url. Instead of passing static arguments from the application profile definition, using this feature you can send dynamic application arguments in a simple way. The only requirement to be met is that the arguments must be url encoded.

To build the url you need:
- The communication protocol (http or https).
- The Thinfinity VirtualUI server domain or IP address, and port.
- The application virtual path.

Build the url like this:

```
protocol://server.address:port/virtualpath/?arguments
```

As an example, we will send the "one", "two" and "three" argument values to the TestArguments program:

```
http://192.168.0.229:6580/TestArguments/?one%20two%20three
```

The following image shows the submit result:
Read more:
- Setting Command Line Arguments in the Application Profile
- Building the URL in Javascript
- Combining Application Profile and URL Command Line Arguments
104

5.4.3

Thinfinity® VirtualUI™

Building the URL in Javascript

When necessary, modern browsers encode URLs automatically, but

it is convenient to
do it by code. The Javascript lines below show how to build the URL used in the
previous example:
var buildURL = function(server, virtualpath, arguments) {
return server + "/" + virtualpath + "/" + encodeURIComponent(arguments.join(' '));
}
var baseURL = "http://192.168.0.229:6580/";
var applicationPath = "TestArguments";
var arguments = ["one", "two", "three"];
var url = buildURL(baseURL, applicationPath, arguments);
Read more:
Setting Command Line Arguments in the Application Profile
Sending Command Line Arguments in the VirtualUI URL
Combining Application Profile and URL Command Line Arguments

© 2018, Cybele Software, Inc.


5.4.4 Combining Application Profile and URL Command Line Arguments

If you have defined command line arguments in the application profile and need to send new arguments to the application by url, don't worry. Thinfinity VirtualUI merges both argument lists, first adding the application profile arguments, and then the arguments that were passed in the url.

Read more:
- Setting Command Line Arguments in the Application Profile
- Sending Command Line Arguments in the VirtualUI URL
- Building the URL in Javascript
5.5 Recording and Playing Sessions

Thinfinity® VirtualUI introduces this feature to help users have a record of their own VirtualUI web-enabled applications' sessions.

In order to enable your application to store sessions of a running VirtualUI-enabled application and then play them, VirtualUI includes a new Recorder component, available for all the supported programming languages. Later, you can play this session from the browser using a special VirtualUI application profile, or you could also implement your own custom session player using the SDK in any of the supported languages.

Read more:
- Recording a Session
- The OnRecorderChanged Event
- Playing Recorded Sessions
- Creating your Own Player
5.5.1 Recording a Session

Set the record filename to the VirtualUI.Recorder component and call its Rec() method to record an application session. The VirtualUI.Recorder object is used in both the recording and the playing process. These operations can be presented, for example, as two buttons in the application: "Rec" and "Stop".

In order to record a session, follow these steps:

1. Choose a file name and location for storing the session. Don't include the extension: sessions will be stored automatically as .dat and .idx.

Set the Filename as follows:

Delphi code:
```delphi
VirtualUI.Recorder.Filename := 'C:\Sessions\VirtualUISession';
```

C++ code:
```cpp
VirtualUI m_VirtualUI = new VirtualUI();
...
...
m_VirtualUI->Recorder()->Filename(L"C:\Sessions\VirtualUISession");
```

C# code:
```csharp
private VirtualUI vui = new VirtualUI();
...
...
vui.Recorder.Filename = "C:\Sessions\VirtualUISession";
```

2. You can specify more than one track for each session. This allows you to, for example, store different application operations. Later, you can play the complete session or a specific track. These tracks are indicated during the sessions recording. Call the Rec() method and pass the track name as an argument in the Label parameter:

Delphi code:
```delphi
VirtualUI.Recorder.Rec('Track 1');
```

C++ code:
```cpp
m_VirtualUI->Recorder()->Rec(L"Track 1");
```

C# code:
```csharp
vui.Recorder.Rec("Track 1");
```
3. In order to stop the recording, call the Stop method:

Delphi code:

```delphi
VirtualUI.Recorder.Stop('Track 1');
```

C++ code:

```cpp
m_VirtualUI->Recorder()->Stop();
```

C# code:

```csharp
vui.Recorder.Stop();
```

Afterwards, you can start another recording with the Rec method. If you keep the same track name, the recording will resume and the session will be stored after the previous session in the same file, in a new track. Change the track name to avoid ending up with two tracks with the same name in the same session recording.

**Read more:**
- [The OnRecorderChanged Event](#)
- [Play Recorded Sessions](#)
- [Create Your Own Player](#)

### 5.5.1.1 The OnRecorderChanged Event

Use the VirtualUI OnRecorderChanged event to listen for any change in the Recorder status.

These are the possible status values:

<table>
<thead>
<tr>
<th>State</th>
<th>It reports the Recorder state:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inactive</td>
<td>0</td>
</tr>
<tr>
<td>Recording</td>
<td>1</td>
</tr>
<tr>
<td>Playing</td>
<td>2</td>
</tr>
</tbody>
</table>

The following examples show how to assign an event handler to this event in the most used languages:

Delphi code:
procedure TMainForm.RecorderChanged(Sender: TObject);
var msg: string;
begin
  msg := '
  case VirtualUI.Recorder.State of
    Inactive: msg := 'Recorder is inactive';
    Playing: msg := 'Recorder is Playing';
    Recording: msg := 'Recording session';
  end;
  ShowMessage(msg);
end;
...
VirtualUI.OnRecorderChanged := Mainform.RecorderChanged;

C++ code:

m_VirtualUI->OnRecorderChanged = RecorderChanged;

void RecorderChanged(void) {
  CString stateStr;
  switch (m_VirtualUI->Recorder()->State()) {
    case Inactive: stateStr = _T("Recorder is inactive"); break;
    case Playing: stateStr = _T("Recorder is Playing"); break;
    case Recording: stateStr = _T("Recording session"); break;
  }
  MessageBox(0, stateStr, L"OnRecorderChanged called", 0);
}

C# code:

private void RecorderChanged(object sender, RecorderChangedEventArgs e)
{
  string message = "";
  switch (vui.Recorder.State) {
    case RecorderState.Inactive:
      message = "Recorder is inactive";
      break;
    case RecorderState.Playing:
      message = "Recorder is playing";
      break;
    case RecorderState.Recording:
      message = "Recording session";
      break;
  }
  MessageBox.Show(message, "OnRecorderChanged called");
}
...
vui.OnRecorderChanged += RecorderChanged;
In order to play a session, create a profile that points to Thinfinity.VirtualUI.Player.exe in the 'ibin' folder of the installation directory. This profile will be available along with the rest of the profiles in the VirtualUI index page:

This is how the VirtualUI Player application looks:

<table>
<thead>
<tr>
<th>DemoAppsHub</th>
<th>easyMoney.Manager</th>
<th>Thinfinity.VirtualUI.Player</th>
<th>VCLTestX</th>
</tr>
</thead>
</table>

Choose the IDX session file to play.
Play, pause and resume the session.
Rewind the session.
Select the speed. Choose 'Normal' to watch the session at the same speed it was recorded. Choose 'Medium' or 'Fast' to play the session at a greater speed.

When you load a session, the screen will show a "Loading..." message. This means the session is ready to play. Press the 'Play' button to start viewing the session. This button will turn to a 'Pause' button. You can also use the 'Rewind' button to start playing the session from the beginning. The numbers on the bottom right corner of the player show the entry being played.

Read more:
- Recording a Session
- The OnRecorderChanged Event
- Creating Your Own Player

5.5.3 Creating your Own Player

You can create your own player application using the SDK. This application won’t need to run in VirtualUI web mode, since it will only use the VirtualUI.Recorder component for playing sessions.

This component will read a previously saved recording to reproduce the same commands that originally were sent by VirtualUI Server to the web browser when running the application. So, an instance of both VirtualUI Player Server and VirtualUI Server—in developer mode—will be automatically launched. You will need also the Development Server URL—by default http://127.0.0.1:6080— to open in the browser.

You will also need:
- A VirtualUI Player Server instance.
- A VirtualUI Server in development mode instance.
- A browser instance where the development Server URL (typically http://127.0.0.1:6080) is open.

The first two are automatically executed when using the Recorder player options.

The playing properties are:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State</strong></td>
<td>It reports the Recorder state. It's Inactive (0) when no action is being made, Recording (1) when a session is being recorded, and Playing (2) when a session is being played.</td>
</tr>
<tr>
<td><strong>Count</strong></td>
<td>Total count of entries in the session file. Each entry is an internal VirtualUI command. The initial position of each track points to one of these file entries.</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>Current entry being played.</td>
</tr>
<tr>
<td><strong>Tracks</strong></td>
<td>List of tracks in the session file. Each track, which is an</td>
</tr>
</tbody>
</table>
IRecTrack interface, has two properties:

<table>
<thead>
<tr>
<th>Name</th>
<th>Original name specified in the Rec() method.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>Entry number in the session file, where the track starts. This value is always between 0 and Count.</td>
</tr>
</tbody>
</table>

First, set the FileName property in the player application to the IDX file name of the session that will be played:

Delphi code:

```delphi
VirtualUI.Recorder.Filename := 'C:\Sessions\VirtualUISession.idx';
```

This will complete the Recorder properties and enable the session to be played, as long as the development URL is open in the browser.

Use the Play method to start playing a session. This method receives two parameters that indicate the initial and final entries to be played.

To reproduce a complete session, you can either send 0 as the beginning and Count as the ending:

Delphi code:

```delphi
VirtualUI.Recorder.Play(0, VirtualUI.Recorder.Count-1);
```

or send -1 in both parameters:

Delphi code:

```delphi
VirtualUI.Recorder.Play(-1, -1);
```

If you wish to play a particular track, send the track range in the Play parameters. The Position property of every track is also the entry number of the beginning, as well the entry number of the ending for the previous track in the list. For the last track, the ending is the Count property.

To stop playing a session before it finishes, use the Stop method:

Delphi code:

```delphi
VirtualUI.Recorder.Stop();
```

Read more:
- [Recording a Session](#)
- [The OnRecorderChanged Event](#)
When using VirtualUI, it is often necessary to be aware of some features and capabilities of the device from which end users are accessing our application.

The VirtualUI class has a property called BrowserInfo, which contains data about the browser being used. This information can assist us in delivering the best possible user experience.

In the list below we will find a detailed reference of the information that BrowserInfo will deliver:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Returns the logged-on username.</td>
</tr>
<tr>
<td>IP Address</td>
<td>Returns the client's ip address.</td>
</tr>
<tr>
<td>Location</td>
<td>Returns the URL of the current application.</td>
</tr>
<tr>
<td>UniqueBrowserId</td>
<td>Identifies an instance of a browser. Each time an end-user opens the application from a different browser window, this identifier will have a different value.</td>
</tr>
<tr>
<td>UserAgent</td>
<td>Returns the browser's User Agent string. This value is very important to identify which browser type is used, if the remote device is a mobile, etc.</td>
</tr>
</tbody>
</table>

An important value to consider is the UserAgent property. When establishing a connection, the browser sends VirtualUI a data string indicating its brand and version, which OS it is running on, and the brand and model of the device that is being used. This value is known as the userAgent. Using this value we can infer, for example, if the user is connecting from a mobile device and then modify some specific feature, layout or functionality of our application to better suit that device.

**Information relative to the available screen space:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrowserWidth</td>
<td>Returns the width of the HTML element containing the VirtualUI viewer, in pixels.</td>
</tr>
<tr>
<td>BrowserHeighth</td>
<td>Returns the height of the HTML element containing the VirtualUI viewer, in pixels.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ScreenWidth</td>
<td>Returns the width of the end-user's device screen, in pixels.</td>
</tr>
<tr>
<td>ScreenHeight</td>
<td>Returns the height of the end-user's device screen, in pixels.</td>
</tr>
<tr>
<td>ScreenResolution</td>
<td>Returns the application screen resolution defined in the application profile.</td>
</tr>
<tr>
<td>ViewWidth</td>
<td>Returns the width of the VirtualUI viewer, in pixels.</td>
</tr>
<tr>
<td>ViewHeight</td>
<td>Returns the height of the VirtualUI viewer, in pixels.</td>
</tr>
<tr>
<td>Orientation</td>
<td>Returns the browser's orientation.</td>
</tr>
</tbody>
</table>

It is also worth mentioning that when using VirtualUI, the desktop size is dynamic. This means that the space available to display the application is correlated to the available space in the browser screen and it is not limited to a fixed desktop size space—as it would happen, for example, in an RDP connection, where the desktop size is defined when the connection is established.

The following C# example shows how to center a window of our application on this virtual desktop as needed, previously controlling its size if it exceeds the available space:

```csharp
using Cybele.Thinfinity;...
...
...
private VirtualUI vui;
public Form1()
{
    InitializeComponent();
    vui = new VirtualUI();
}
private void CenterThisWindow(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Normal;
    if (vui.BrowserInfo.ViewHeight < (this.Height + 20))    {
        this.Height = vui.BrowserInfo.ViewHeight - 20;
    }
    if (vui.BrowserInfo.ViewWidth < (this.Width + 20))    {
        this.Width = vui.BrowserInfo.ViewWidth - 20;
    }
```
}

    this.Top = (vui.BrowserInfo.ViewHeight - this.Height) / 2;
    this.Left = (vui.BrowserInfo.ViewWidth - this.Width) / 2;
}


5.7 Change Browser Behavior Using ClientSettings

<ClientSettings> is an additional interface available in the Thinfinity VirtualUI Library that allows developers to remotely and programmatically configure particular browser settings. These settings are related to the cursor visibility and some specific touch action behaviors.

There is also a Javascript Clientsettings object version. [Learn how to use it](#).

The table below shows a detailed reference of the <ClientSettings> interface current properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>CursorVisible</td>
<td>Used to set the mouse pointer to 'show' or 'hide'.</td>
</tr>
<tr>
<td>Type: Boolean</td>
<td>Default value: True</td>
</tr>
<tr>
<td>Available in all Library Versions.</td>
<td></td>
</tr>
</tbody>
</table>

MouseMoveGestureStyle

- Defines whether the mouse movement is read as relative or absolute in touch devices.
- Type: Enum
- Values:
  - MM_STYLE_RELATIVE = 0
  - MM_STYLE_ABSOLUTE = 1
- Default value: MM_STYLE_RELATIVE
- Available in all Library Versions.

MouseMoveGestureAction

- Indicates if the dragging should be interpreted as scrolling or turning of the mouse wheel, affecting only the component on which the action is triggered.
- Type: Enum
- Values:
  - MM_ACTION_MOVE = 0
  - MM_ACTION_WHEEL = 1
- Default value: MM_ACTION_MOVE
- Available in all Library Versions.

UseViewportSize

- Indicates which measure values will be used to define the application's screen size.
- Type: Boolean
- True: use the viewport size
- False: use a window screen based size.
- Default value: False
- Available only in Javascript.

DockMenu

- Manages the visibility of the bottom menu and its options.
- Available only in Javascript.
| **Enabled** | Boolean. Enables/disables the complete Dock Menu. **Type:** Boolean **Default value:** True |
| **Pinned** | Indicates when the DockMenu is pinned (visible, fixed) or unpinned (with autohide) **Type:** Boolean **Default value:** False |
| **Items** | Name/value pairs item collection. |
| **WindowList** | Enables/disables the "Windows list" option. **Type:** Boolean **Default value:** True |
| **ErrorReporting** | Enables/disables the "Error Reporting" option. **Type:** Boolean **Default value:** True |
| **Keyboard** | Enables/disables the "Keyboard" option (mobile only). **Type:** Boolean **Default value:** True |
| **FullScreen** | Enables/disables the "Full Screen" option (mobile only, if supported). **Type:** Boolean **Default value:** True |

As mentioned previously, these properties were created to provide a better end-user experience when using touch devices.

When using a touch device, it is unusual for a mouse pointer to be displayed. Instead, finger gestures are interpreted by the device as different commands, acting on the element or elements that are located where the gesture is made.

But this is not always the best way, especially when the applications were not designed for these gestures. Most of the times, the best solution to this issue is to show the pointer and simulate the mouse actions using the captured gestures.

The properties that we have presented above will help us define the ideal settings to enjoy the best possible and intuitive user experience for our application.

The following example shows how to recognize a touch or mobile device, and to change the pointer behavior when it's detected. Warning: The detection list is not complete:

```java
vui = new VirtualUI();
userAgent = vui.BrowserInfo.UserAgent.ToLower();
```
if ((userAgent.IndexOf("touch") != -1) || (userAgent.IndexOf("mobile") != -1) ||
    (userAgent.IndexOf("iphone") != -1) || (userAgent.IndexOf("ipod") != -1) ||
    (userAgent.IndexOf("ipad") != -1) || (userAgent.IndexOf("android") != -1) ||
    (userAgent.IndexOf(" cros ") != -1)) {
    vui.ClientSettings.MouseMoveGestureStyle = MouseMoveGestureStyle.MM_STYLE_ABSOLUTE;
}

Read more about using ClientSettings from Javascript.
5.8 End-User Authentication

Thinfinity VirtualUI allows you to protect the published applications by applying Active Directory (AD) objects to each one of them. In order to be able to see and execute an application in the Thinfinity VirtualUI environment, either the application must have anonymous access or the end-user must provide credentials that satisfy the AD objects assigned to that application.

Thinfinity VirtualUI lets you activate one or more authentication methods, allowing for the mapping of credentials to AD objects, which in turn will grant access to the applications that had the same AD objects applied. Also, the end-user identification is passed on to the application in order to allow a Single Sign-On.

Read the following sections to learn how to implement the available authentication methods in VirtualUI web-enabled applications.

Read more:
- Entering Credentials
- Processing End-User Credentials
- Authentication Methods
5.8.1 Entering Credentials

Thinfinity VirtualUI allows you to enable one or more authentication methods at a time. There are two possible ways to ask for credentials:

- Using the standard Web Browser authentication dialog
- Showing a log-in page

The *Standard Web Browser Authentication* (aka *Basic Authentication*) is, in the context of an HTTP/HTTPS transaction, a method for the HTTP User Agent (the Web Browser) to provide end-user credentials when a request is started. The standard Web Browser authentication dialog is provided by each Web Browser and it looks like this:

![Standard Web Browser Authentication dialog](image)

This dialog is available when you use only one of the authentication access methods that require user and password: *Windows Logon*, *RADIUS* or *External DLL*.

Also, you can use a login page. The login page provided (login.html) was created to dynamically show all configured authentication methods in your Thinfinity VirtualUI Server. Every login option will be present only if the proper authentication method is configured.

For example, if only the 'Windows Logon' method is configured, the page will look like this:
Enter credentials to sign in

Username
Password

→ Sign in

But if you enable all predefined methods, your login will show something like this:

Sign in or select an option

<table>
<thead>
<tr>
<th>Login with Google</th>
<th>Username</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login with Facebook</td>
<td>Password</td>
</tr>
<tr>
<td>Login with LinkedIn</td>
<td></td>
</tr>
<tr>
<td>Login with Dropbox</td>
<td></td>
</tr>
</tbody>
</table>

or

| Use Windows Logon | Use Windows Logon |
| Use Radius        | Use My own authentication method |

You can modify this page to adapt it to your branding and/or integration needs. Check out the login.html section to learn how.

Read more:
- Processing End-User Credentials
- Authentication Methods
- login.html
5.8.2 Processing End-user Credentials

Each published application without anonymous access requires the assignment of one or more Active Directory objects which define the users that can see and execute it.

Thinfinity VirtualUI implements a mapping mechanism to transform end-user credentials to AD objects. Only applications that match these AD objects will be granted access to the end-user.

When you enable an authentication method, you must add the mapping rules that will allow you to link the user ID with an AD object. This is done by specifying an external user ID mask and its linked AD objects.

Depending on the selected authentication method, Thinfinity VirtualUI uses the identification provided provided by the user to scan the mapping rule list and obtains the associated AD objects. If the matching process returns one or more AD objects, all applications with the same AD object are enabled to be seen and accessed by the end-user.

Read more:
- Entering Credentials
- Authentication Methods
5.8.3 Authentication Methods

Thinfinity VirtualUI allows you to use the following authentication methods.

**Windows Logon**
This option enables Active Directory credentials. This method is enabled by default.

**RADIUS**
Remote Authentication Dial-In User Service (RADIUS) is a networking protocol and software that provides centralized Authentication, Authorization, and Accounting (AAA) management for users who remotely connect to a network service, in this case the Thinfinity VirtualUI Server.

**OAuth 2.0**
OAuth 2.0 (or OAuth/2) is an open standard for authorization and authentication, commonly used as a way for Internet users to log into third party websites using their social network (Facebook, Google, LinkedIn, etc.) account without exposing their password.

**External DLL**
A custom authentication method implemented by you or a third party with our authentication API and referenced in the Thinfinity VirtualUI server.

**Anonymous Authentication**
You can also allow users to access applications anonymously. When this access is combined with other authentication methods, the anonymous access applications will be shown along with a 'Sign in' link for users to enter credentials.

All these methods will be enabled and configured in the Thinfinity VirtualUI Server Manager: choose the methods in the 'Authentication' tab and configure the mapping in the 'Mappings' subtab.

Read more:
- Windows Logon
- RADIUS
- OAuth 2.0
- External DLL
- Entering Credentials
- Processing End-User Credentials

5.8.3.1 Windows Logon

Windows Logon means that the end-user will have to enter Windows Activate Directory credentials in order to gain access to a set of applications defined in Thinfinity VirtualUI.

The profiles matching the credentials provided in their the 'Permissions' tab will be the profiles shown to the authenticated user, along with those with the 'Allow Anonymous Access' option checked in their 'Permissions' tab.

Windows Logon is enabled by default in the 'Authentication' tab of the Thinfinity
VirtualUI Server Manager. Toggle its availability as an authentication method by checking or unchecking it.

- Allow anonymous access
- Use standard browser authentication dialog

Read more:
- RADIUS
- OAuth 2.0
- External DLL
5.8.3.2 RADIUS

RADIUS as an authentication method means that the user will have to enter their RADIUS credentials in order to gain access to a set of applications defined in Thinfinity VirtualUI.

RADIUS can be added in the 'Authentication' tab of the Thinfinity VirtualUI Server Manager. Toggle its availability as an authentication method by checking or unchecking it.

When you add RADIUS as an authentication method you will be required to provide the RADIUS account relevant information.
The user definition is completed through the mapping between the user ID returned by RADIUS and a user registered for this Authentication ID Mask. The RADIUS credentials are mapped to Active Directory Objects in the 'Mappings' tab. Those Active Directory objects should satisfy the permission access rules of the applications that they are expected to get access to.
Read more:

- Windows Logon
- OAuth 2.0
- External DLL
5.8.3.3 OAuth 2.0

OAuth 2.0 is a standard authentication method used mostly in social web sites. The user will have to enter their OAuth 2.0 (Facebook, Dropbox, LinkedIn, Google or other) credentials in order to gain access to a set of applications defined in Thinfinity VirtualUI.

An OAuth 2.0 authentication method can be added in the 'Authentication' tab of the Thinfinity VirtualUI Server Manager. Toggle their availability as an authentication method by checking or unchecking them.

When you add an OAuth 2.0 method you will be required to provide the relevant information. Check the complete reference in the OAuth 2.0 Authentication Method Settings. This information is also reflected in the OAuth2Models.ini file, distributed...
with the installation.

The user definition is completed through the mapping between the user ID returned by the selected OAuth 2.0 method (in the examples mentioned it's always the email) and a user registered for this Authentication ID Mask. The OAuth 2.0 method’s credentials are mapped to Active Directory Objects in the 'Mappings' tab. Those Active Directory objects should satisfy the permission access rules of the applications that they are expected to get access to.

Read more:
- OAuth2Models.ini
- CSS for SSO options
5.8.3.3.1 OAuth2Models.ini

OAuth2Models.ini is a file is distributed with the Thinfinity VirtualUI installation that has all the information of the default available logins:

```
[Google]
ClientID =
ClientSecret =
AuthorizationParameters = scope=https://www.googleapis.com/auth/
userinfo.email&approval_prompt=auto
AuthorizationURL = https://accounts.google.com/o/oauth2/auth
TokenValidationURL = https://accounts.google.com/o/oauth2/token
ProfileValidationURL = https://www.googleapis.com/oauth2/v1/userinfo
UsernameField = email

[Facebook]
ClientID =
ClientSecret =
AuthorizationParameters = scope=email
TokenValidationURL = https://graph.facebook.com/oauth/access_token
ProfileValidationURL = https://graph.facebook.com/me?
UsernameField = email

[LinkedIn]
ClientID =
ClientSecret =
AuthorizationParameters =
state=HIJK98sDT88jnS23S&scope=r_emailaddress
TokenValidationURL = https://www.linkedin.com/uas/oauth2/accessToken
ProfileValidationURL = https://api.linkedin.com/v1/people/~:(emailAddress)?format=json
UsernameField = emailAddress

[Dropbox]
ClientID =
ClientSecret =
AuthorizationURL = https://www.dropbox.com/1/oauth2/authorize
AuthorizationParameters =
TokenValidationURL = https://api.dropboxapi.com/1/oauth2/token
ProfileValidationURL = https://api.dropboxapi.com/1/account/info
UsernameField = email
```

Use this file as a template and edit it in order to add new authentication methods or ask your authentication provider for different data.

Read more:
- OAuth 2.0
5.8.3.3.2 CSS for SSO Options

In the login.css file, included in the 'web\css' folder of the Thinfinity VirtualUI installation, you will find the style for the login buttons.

```css
#google { background-color:#4285F4; }  
#google .imgbtn { background-image: url(../images/sso/google.png); }

#facebook { background-color:#2f4e9e; }  
#facebook .imgbtn { background-image: url(../images/sso/facebook.png); }

#yahoo { background-color:#6839af; }  
#yahoo .imgbtn { background-image: url(../images/sso/yahoo.png); background-size: 30px; }

#linkedin { background-color:#00A0DC; }  
#linkedin .imgbtn { background-image: url(../images/sso/linkedin.png); }

#dropbox { background-color:#007ee6; }  
#dropbox .imgbtn { background-image: url(../images/sso/dropbox.png); background-size: 30px; }
```

You can change the logo and/or background color of the login buttons for the Oauth 2.0 authentication methods.

Each pair of entries corresponds to one authentication method. The ID (#google, #facebook) must match the Virtual Path established in the Oauth 2.0 Authentication Method Settings. With the installation, these parameters are matching by default, but make sure to change it in both places if you do.

The first line of each pair defines the button color, and the second one the button image.

**Read more:**
- Entering Credentials
- Processing End-User Credentials
- Authentication Methods
- Windows Logon
- RADIUS
- OAuth 2.0
- OAuth2Models.ini
- External DLL
- Authentication API
5.8.3.4 External DLL

Thinfinity VirtualUI allows you to integrate your own custom authentication method. In order to do this, use the Thinfinity VirtualUI Authentication API.

The External DLL authentication method can be added in the 'Authentication' tab of the Thinfinity VirtualUI Server Manager. Toggle its availability as an authentication method by checking or unchecking it.

When you add an External DLL authentication method you will be required to reference the .dll in the 'External DLL' option. Check the complete reference in the External DLL Authentication Method Settings.

The user definition is completed through the mapping between the user ID returned by the external DLL and an Active Directory Objects mapped for this Authentication ID Mask.
Read more:
- Authentication API
5.8.3.4.1 Authentication API

Thinfinity VirtualUI provides you with an API that you can use to develop your own authentication method and integrate it with VirtualUI.

Choose the code sample of your language of preference and add it to your implementation:

**Delphi:**

```delphi
function ValidateUser(
    const UserName, Password, Metadata: PWideChar;
    SecurityRole, WinUser, WinPass, CustomData: PWideChar;
    var Handled: Boolean): Cardinal; stdcall;
```

**Input:**

<table>
<thead>
<tr>
<th>Username &amp; Password</th>
<th>The credentials that you are trying to validate with the external authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata</td>
<td>A JSON with the remote browser/user information: URL, IP, Cookie UBRWID and the product's name</td>
</tr>
</tbody>
</table>

**Output:**

<table>
<thead>
<tr>
<th>SecurityRole</th>
<th>Specifies the Windows mapping of the authenticated user (UserName and Password). This SecurityRole can either be a Windows user or group, and it will be used to check which profiles it has access to</th>
</tr>
</thead>
<tbody>
<tr>
<td>WinUser, WinPass</td>
<td>(optional) Credentials of a mapped Windows user. Will be used to run the application instance.</td>
</tr>
<tr>
<td>CustomData</td>
<td>(optional) Data for passing on to the application</td>
</tr>
<tr>
<td>Handled</td>
<td>Returns whether the login could be handled by the DLL.</td>
</tr>
</tbody>
</table>

**C++:**

```c++
```
### Input:

<table>
<thead>
<tr>
<th>lpUserName &amp; lpPassword</th>
<th>The credentials that you are trying to validate with the external authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpMetadata</td>
<td>A JSON with the remote browser/user information: URL, IP, Cookie UBRWID and the product's name</td>
</tr>
</tbody>
</table>

### Output:

<table>
<thead>
<tr>
<th>lpSecurityRole</th>
<th>Specifies the Windows mapping of the authenticated user (UserName and Password). This SecurityRole can either be a Windows user or group, and it will be used to check which profiles it has access to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpWinUser, lpWinPass</td>
<td>(optional) Credentials of a mapped Windows user. Will be used to run the application instance.</td>
</tr>
<tr>
<td>lpCustomData</td>
<td>Data for passing on to the application</td>
</tr>
<tr>
<td>pHandled</td>
<td>Returns whether the login could be handled by the DLL.</td>
</tr>
</tbody>
</table>

### C#:

```csharp
[DllImport("ValidateUser", CallingConvention = CallingConvention.StdCall)]
[return: MarshalAs(UnmanagedType.I4)]
public static Int32 ValidateUser(
    [In, MarshalAs(UnmanagedType.LPWStr)] string lpUserName,
    [In, MarshalAs(UnmanagedType.LPWStr)] string lpPassword,
    [In, MarshalAs(UnmanagedType.LPWStr)] string lpMetadata,
    [In, Out, MarshalAs(UnmanagedType.LPWStr)] StringBuilder lpSecurityRole,
    [In, Out, MarshalAs(UnmanagedType.LPWStr)] StringBuilder lpWinUser,
    [In, Out, MarshalAs(UnmanagedType.LPWStr)] StringBuilder lpWinPass,
    [In, Out, MarshalAs(UnmanagedType.LPWStr)] StringBuilder lpCustomData,
    [Out] bool pHandled);
```

### Input:

| lpUserName & lpPassword | Specifies the Windows mapping of the authenticated user (UserName and Password). This SecurityRole can either be a Windows user or group, and it will be used to check which profiles it has access to |

© 2018, Cybele Software, Inc.
### IpMetadata
A JSON with the remote browser/user information: URL, IP, Cookie UBRWID and the product's name

### Output:

<table>
<thead>
<tr>
<th>IpSecurityRole</th>
<th>The authenticated username</th>
</tr>
</thead>
<tbody>
<tr>
<td>IpWinUser, IpWinPass</td>
<td>(optional) Credentials of a mapped Windows user. Will be used to run the application instance.</td>
</tr>
<tr>
<td>IpCustomData</td>
<td>Data for passing on to the application</td>
</tr>
<tr>
<td>pHHandled</td>
<td>Returns whether the login could be handled by the DLL.</td>
</tr>
</tbody>
</table>

### Read more:
- [External DLL](#)
5.9 Virtualization

In Thinfinity VirtualUI, each application instance may run under a unique Windows User ID or under a shared one. In the first case, the application may access private folders and private registry entries but, as most applications provide their own credential system (usually based on a database), it is a common practice to run the application instance under a single shared Windows User ID. In this case, all application instances share the same user profile’s folders and the same “current user” key in the registry.

Additionally, in both cases, all users may have access to shared folders that it might be desirable to hide from them.

Thinfinity VirtualUI addresses these potential drawbacks by providing the File System and Registry Virtualization feature.

The main purposes of the Registry and File System virtualization are:

- To show only relevant folders.
- To provide private folders based on the application's credential system.
- To provide registry keys based on the application's credential system.

Read more:
- File System Virtualization
- Registry Virtualization
- Implementing Virtualization in Your Application

5.9.1 File System Virtualization

File System Virtualization helps developers publish only relevant shared folders and provide authentication-based private folders. Additionally, shared folders can be redirected to safer places in the disk and/or be write protected.

Virtualization involves two steps. The first one is the File System mapping design, when the developer creates the relationship between real paths and the redirected path tree that the end-user will see and access. The second one, the inclusion of the created mapping definition into the application program.

The file system mapping definition is a set of one to one redirection rules that link a real folder to the redirected folder path.

There are two types of redirection:

- Shared: for all connected users. It can be defined as readonly.
- Private: for a single identified user.

The configuration is done through the The File System Virtualization Editor, which will be installed with the Standard or server component installation.
5.9.1.1 Creating File System Redirections

In order to add a path in the Virtualization Editor, right click on the main area of the screen and choose 'New Path...'.

This will bring up the 'Edit Directory' dialog.
### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>Choose 'Shared' if you want this mapping to apply to all users. Choose 'Private' if you only want this mapping to apply to a single user.</td>
</tr>
<tr>
<td>Read Only</td>
<td>Only available for Shared Mode. Check this option to make the mapped directory read only for the user.</td>
</tr>
<tr>
<td>Directory</td>
<td>Choose the directory you want to show.</td>
</tr>
<tr>
<td>Redirect to</td>
<td>Choose the physical path where the directory contents will be stored during the user session.</td>
</tr>
</tbody>
</table>

Paths can also be expressed using the environment variables defined in Windows: %public%, %home%, %documents%, etc.

For example, you can redirect %userprofile% to %userprofile%\users as a private redirection; "user1"'s files will therefore be stored in `c:\users\username\users\user1`

Or you can map c:\temp to %userprofiles%\temp as a shared redirection.

**Editing Paths**

In order to edit a path, right click on it and choose 'Edit Path'.

© 2018, Cybele Software, Inc.
You will then be able to edit the path in the 'Edit Directory' dialog.

**Deleting Paths**

In order to delete a path, right click on it and choose 'Delete Path'.
Read more:
- The .vidr Files
- Testing the File System Virtualization
- Implementing Virtualization in Your Application
5.9.1.2 The .vdir Files

Once you are done adding, editing and/or deleting paths, save this configuration by selecting 'File' - 'Save' in the main menu of the Filesystem Virtualization Editor. This configuration will be stored in a vdir file. The Filesystem Virtualization Editor allows you to create different vdir files to be used in your application.

Read more:
- Creating File System Redirections
- Testing the File System Virtualization
- Implementing Virtualization in Your Application
5.9.1.3 Testing the File System Virtualization

You can test the Filesystem virtualization outside VirtualUI (the virtualization service must be running, though). In order to test the Filesystem virtualization, choose the 'File' - 'Test' option from the main menu. This will bring up the 'Launch Application' dialog:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Type in a testing user name. Later on, you can test using another user and verify that the files stored for each user are isolated.</td>
</tr>
<tr>
<td>Program</td>
<td>Choose a program with which to test the Filesystem virtualization. It has to be a program capable of opening files.</td>
</tr>
</tbody>
</table>

After selecting a user and a program to test, press the 'Launch' button. When you access the Windows file system through this application, you should only be seeing the mapped directories, by their original names.

Read more:
- Creating File System Redirections
- The .vidr Files
- Implementing Virtualization in Your Application

5.9.2 Registry Virtualization

Registry Virtualization allows developers to store end-user information in the Windows Registry in a secure and isolated way and to provide authentication-based registry entries. Additionally, shared registry entries can be redirected to safer places in the Windows Registry.

Similarly to File System Virtualization, Registry Virtualization also involves two steps. The first one is the mapping definition, when the developer creates the relationship between an original registry entry and a redirected registry entry. The second one, the inclusion of the created mapping definition into the application program.
The registry mapping definition is a set of one to one redirection rules that link a real entry to the redirected entry.

There are two types of redirection:
- Shared: for all connected users.
- Private: for a single identified user.

Private redirection entries include the identified user ID appended to the redirected entry.

The configuration is done through the Registry Virtualization Editor, which will be installed with the Standard or server component installation.

**Read more:**
- Creating Registry Redirections
- The .vreg Files
- Testing Registry Virtualization
- File System Virtualization
- Implementing Virtualization in Your Application

### 5.9.2.1 Creating Registry Redirections

In order to add a registry redirection, right click on one of the Server trees and choose 'Add Redirection'.

<table>
<thead>
<tr>
<th>Add Redirection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Redirection</td>
</tr>
<tr>
<td>Delete Redirection</td>
</tr>
</tbody>
</table>
This will bring up the 'Edit Key Redirection' dialog:

![Edit Key Redirection dialog]

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode</strong></td>
<td>Choose 'Shared' if you want this key redirection to apply to all users.</td>
</tr>
<tr>
<td></td>
<td>Choose 'Private' if you only want this key redirection to apply to a single user.</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Choose the source registry key you want to redirect.</td>
</tr>
<tr>
<td><strong>Destination</strong></td>
<td>Choose a registry destination key.</td>
</tr>
</tbody>
</table>

**Editing Redirections**

In order to edit a key redirection, right click on it and choose 'Edit Redirection'.

© 2018, Cybele Software, Inc.
You will then be able to edit the redirection in the 'Edit Key Redirection' dialog.

**Deleting Redirections**

In order to delete a redirection, right click on it and choose 'Delete Redirection'.

**Read more:**
- [Testing Registry Virtualization](#)
- [The .vreg Files](#)
- [File System Virtualization](#)
- [Implementing Virtualization in Your Application](#)
5.9.2.2 The .vreg Files

Once you are done adding, editing and/or deleting redirections, save this configuration by selecting 'File' - 'Save' in the main menu of the Registry Virtualization Editor. This configuration will be stored in a vreg file. The Registry Virtualization Editor allows you to create different vreg files to be used with the product.

Read more:
- Creating Registry Redirections
- Testing Registry Virtualization
- File System Virtualization
- Implementing Virtualization in Your Application
5.9.2.3 Testing Registry Virtualization

In order to test the registry virtualization, choose the 'File' - 'Test' option from the main menu. This will bring up the 'Launch Application' dialog:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Type in a testing user name. Later on, you can test using another user and verify that the registry changes stored for each user are isolated.</td>
</tr>
<tr>
<td>Program</td>
<td>Choose a program with which to test the registry virtualization. It has to be a program that uses the registry.</td>
</tr>
</tbody>
</table>

After selecting a user and a program from the test, press the 'Launch' button. This will open the selected application and create a unique key for the user in the redirected key. All the activity of this application in the registry for the source key will be reflected in the redirected key.

Read more:
- Creating Registry Redirections
- File System Virtualization
- Implementing Virtualization in Your Application

5.9.3 Implementing Virtualization in Your Application

In order to apply the virtualization filters created with the Filesystem Virtualization Editor or the Registry Virtualization Editor, you have to add some code to your application.

After `virtualui.Start()`, load and apply the created filters with the current user credentials

**Delphi:**

```delphi
virtualui.RegistryFilter.User := 'user identification';
virtualui.RegistryFilter.CfgFile := 'file.vreg';
```
virtualui.RegistryFilter.Active := true;
virtualui.FilesystemFilter.User := 'user identification';
virtualui.FilesystemFilter.Active := true;

C#:

virtualui.Start();
virtualui.RegistryFilter().User = "user identification";
virtualui.RegistryFilter().CfgFile = "file.vreg";
virtualui.RegistryFilter().Active = True;
virtualui.FilesystemFilter().User = "user identification";
virtualui.FilesystemFilter().CfgFile = "file.vdir";
virtualui.FilesystemFilter().Active = True;

C++:

virtualui->Start();
virtualui->RegistryFilter()->User = "user identification";
virtualui->RegistryFilter()->CfgFile = "file.vreg";
virtualui->RegistryFilter()->Active = True;
virtualui->FilesystemFilter()->User = "user identification";
virtualui->FilesystemFilter()->CfgFile = "file.vdir";
virtualui->FilesystemFilter()->Active = True;

In order to disable registry or filesystem filtering, set the **Active** property to false.

**Read more:**
- File System Virtualization
- Registry Virtualization

### 5.10 OEM Licensing

If your product qualifies for OEM licensing, we will provide you with a Thinfinity VirtualUI OEM Licensing Library that enables OEM-licensed Companies to create and revoke end-user Thinfinity VirtualUI licenses.

There are two prerequisites:
- The application that uses the OEM Licensing Library (Thinfinity.VirtualUI.OEM.dll) must be digitally signed.
- You must have a valid OEM key.

To get an OEM key, you must provide Cybele Software with a digital certificate's thumbprint. On each licensing request, three parameters will be used to validate the requestor:
- The OEM-licensed Company's e-mail.
- The provided OEM key.
- The digital certificate's fingerprint, taken from the digital certificate that signs the executable making the OEM library call.
The generated OEM key and the digital certificate's thumbprint, along with your customer identification (e-mail), will be used to securely validate the licensing requests your application will make.

**Read more:**
- [How to Create and Revoke Licenses](#)
- [Creating Licenses](#)
- [Revoking Licenses](#)
5.10.1 How to Create and Revoke Licenses

ViSThe OEM library provides two functions to create a license and two functions to revoke a license:
- CreateLicenseW (Unicode version) and CreateLicenseA (ANSI version).
- RevokeLicenseW (Unicode version) and RevokeLicenseA (ANSI version).

In order to test the license creation and activation process, you can enable the SandboxMode in the OEM.ini file.

```
[LICENSE]
SandboxMode = True
```

The OEM.ini file should be placed in both the bin32 and bin64 paths under the Thinfinity VirtualUI installation folder and in the user's application executable file folder.

The software will first look for the file in the executable's directory (either the application's or any server's). If it doesn't find it there, there are two alternatives:
- For the virtualizer, which is in ibin, it will search in bin64 and, afterwards, in bin32
- For the rest of the cases, it searches for it in the parent directory. That is, if something runs in bin32 or bin64 it can use an oem.ini from their parent directory.

Adding External Methods

In a Delphi program we must first declare the external functions in their ANSI or Unicode version.

ANSI:

```pascal
function CreateLicenseA(const email, oemkey, customerid: PAnsiChar; units: DWORD; serial: PAnsiChar): integer; stdcall;
external 'Thinfinity.VirtualUI.OEM.dll';

function RevokeLicenseA(const email, oemkey, serial: PAnsiChar): integer; stdcall;
external 'Thinfinity.VirtualUI.OEM.dll';
```

Unicode:

```pascal
function CreateLicenseW(const email, oemkey, customerid: PWideChar; units: DWORD; serial: PWideChar): integer; stdcall;
external 'Thinfinity.VirtualUI.OEM.dll';

function RevokeLicenseW(const email, oemkey, serial: PWideChar): integer; stdcall;
external 'Thinfinity.VirtualUI.OEM.dll';
```
In order to access external library functions in a **C#** program, please declare:

```csharp
[DllImport("Thinfinity.VirtualUI.OEM.dll",
    CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Unicode)
private static extern int CreateLicenseW(string email, string oemkey,
    string customerid, int units, StringBuilder serial);

[DllImport("Thinfinity.VirtualUI.OEM.dll",
    CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Unicode)
private static extern int RevokeLicenseW(string email, string oemkey, string serial);
```

**Read more:**
- Creating Licenses
- Revoking Licenses

### 5.10.1.1 Creating Licenses

In order to create a new Thinfinity VirtualUI license you must use the corresponding `CreateLicense` method with the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>in</td>
<td>OEM-licensed Company´s e-mail</td>
</tr>
<tr>
<td>OEMKey</td>
<td>in</td>
<td>OEM Key.</td>
</tr>
<tr>
<td>CustomerID</td>
<td>in</td>
<td>Arbitrary customer identification for the new license. This can be an e-mail address, a serial number or any other customer identification.</td>
</tr>
<tr>
<td>Units</td>
<td>in</td>
<td>Number of units to enable in this license. Each unit contains a predefined number of concurrent users. If your OEM license has 5 users per unit, and you assign 2 units to the product license, the customer will have 10 concurrent users in total.</td>
</tr>
<tr>
<td>Serial</td>
<td>out</td>
<td>Serial number of the successfully generated license.</td>
</tr>
</tbody>
</table>

A call to the `CreateLicense` function will return one of the following error codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error. The &quot;Serial&quot; parameter will contain the generated serial number.</td>
</tr>
</tbody>
</table>
The following example shows how to create a license using **Delphi**. Please note that you must replace `CreateLicenseW` by `CreateLicenseA` when using ANSI chars.

```delphi
procedure CreateLicense;
var
  result: Integer;
  serial: WideString;
begin
  serial := StringOfChar(' ', 128);
  result := CreateLicenseW(
    PChar('OEM_EMAIL'),
    PChar('OEM_KEY'),
    PChar('CUSTOMER_ID'), UNIT_COUNT,
    PWideChar(serial));
  // Check result
end;
```

And this is how you create it using **C#**:

```csharp
private void CreateLicense()
{
  StringBuilder sn = new StringBuilder(128);
  int result = CreateLicenseW("OEM_EMAIL", "OEM_KEY",
    "CUSTOMER_ID", UNIT_COUNT, sn);
  string serial = "";
  // Check result
  if (result == 0)
  {
    serial = sn.ToString().Trim();
  }
  else
```
Note: In all these examples, please replace 'OEM_EMAIL' and 'OEM_KEY' with your Company's OEM registration, 'CUSTOMER_ID' with the Customer identification and 'UNIT_COUNT' with the desired value.

Read more:
- How to Create and Revoke Licenses
- Revoking Licenses
5.10.1.2 Revoking Licenses

The **RevokeLicense** function is used to revoke a previously generated license. These are its parameters:

<table>
<thead>
<tr>
<th>Email</th>
<th>in</th>
<th>OEM-licensed Company's e-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>OEMKey</td>
<td>in</td>
<td>OEM License Key.</td>
</tr>
<tr>
<td>Serial</td>
<td>in</td>
<td>Serial number to revoke.</td>
</tr>
</tbody>
</table>

This function will return one of the following error codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error. The license was revoked.</td>
</tr>
<tr>
<td>-1</td>
<td>General error in license server.</td>
</tr>
<tr>
<td>-2</td>
<td>Incorrect Email or OEMKey.</td>
</tr>
<tr>
<td>-3</td>
<td>Incorrect Serial number.</td>
</tr>
<tr>
<td>-4</td>
<td>Incorrect Serial number.</td>
</tr>
<tr>
<td>-5</td>
<td>General error. Cannot revoke license.</td>
</tr>
<tr>
<td>-10</td>
<td>Invalid response received from license server.</td>
</tr>
<tr>
<td>-20</td>
<td>Malformed response received from license server.</td>
</tr>
<tr>
<td>-50</td>
<td>Error in HTTP request.</td>
</tr>
<tr>
<td>-100</td>
<td>General error in library.</td>
</tr>
<tr>
<td>-101</td>
<td>Application invalid: not digitally signed.</td>
</tr>
<tr>
<td>&gt;200</td>
<td>HTTP status code.</td>
</tr>
</tbody>
</table>

Here's how to revoke a license using **Delphi**:

```delphi
procedure RevokeLicense;
var
  result: Integer;
begin
  result := RevokeLicenseW(PChar('OEM_EMAIL'), PChar('OEM_KEY'), PChar('SERIAL_NUMBER'));
end;
```
And here's how to revoke a license using **C#**: 

```csharp
private void RevokeLicense() {
    int result = RevokeLicenseW("OEM_EMAIL", "OEM_KEY", "SERIAL_NUMBER");
}
```

**Note:** Please replace 'OEM_EMAIL' and 'OEM_KEY' with your Company's OEM registration information, and replace 'SERIAL_NUMBER' with the Serial to revoke.

**Read more:**
- [How to Create and Revoke Licenses](#)
- [Creating Licenses](#)
5.11 Silent Install Options

The VirtualUI installation can be run in 'silent' mode, that is, without the need for user interaction. This can be useful if you are a system administrator and you want to automate the VirtualUI installation or if you are deploying it over your local network. Also, if you have an OEM Licensing model you will probably use a silent install.

**Thinfinity VirtualUI Line Switches**

In order to perform a silent installation, use this command line:

c:\Thinfinity_VirtualUI_Setup_x64.exe /s /v/qn

These are additional command line switches that you can pass on to the setup file:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFMX_SERVER_MODE</td>
<td>Determines whether the server components will be installed</td>
<td>TRUE</td>
</tr>
<tr>
<td>VFMX_DEV_MODE</td>
<td>Determines whether the development components will be installed</td>
<td>TRUE</td>
</tr>
<tr>
<td>VIRTUALIZATION</td>
<td>Determines whether the Virtualization components will be installed. In order for this option to be enabled, VFMX_SERVER_MODE cannot be FALSE.</td>
<td>FALSE</td>
</tr>
<tr>
<td>STDLBMODE</td>
<td>Values: - LBMODE: Load balancing mode - STDMODE: Standard mode.</td>
<td>STDMODE</td>
</tr>
<tr>
<td>SM_TYPE</td>
<td>Values: - SM_Complete : Installs Server and Gateway components</td>
<td></td>
</tr>
</tbody>
</table>
- SM_Broker: Installs only Server components
- SM_Gateway: Installs only Gateway components.

Note: In order for this option to be enabled, `STDLBMODE` has to be set to `LBMODE`.

<table>
<thead>
<tr>
<th>EMAIL</th>
<th>Complete this variable with your registration email. Also make sure to include the <code>SERIAL</code> parameter in order for the registration to work.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERIAL</td>
<td>Complete this variable with your registration serial. Also make sure to include the <code>EMAIL</code> parameter in order for the registration to work.</td>
</tr>
</tbody>
</table>

The `VFMX_SERVER_MODE`, `VFMX_DEV_MODE` and `VIRTUALIZATION` parameters correspond to these installation wizard options:

The `STDLBMODE` and `SM_TYPE` parameters correspond to these installation wizard options:
The default installation will install Thinfinity VirtualUI Server in standard mode and the development components, but not the virtualization components.

**Examples**

- Installing Thinfinity VirtualUI in Standard Mode with Virtualization Components and without Development Environment:

  c:\Thinfinity_VirtualUI_v2_Setup_x64.exe /s /v"/qn VIRTUALIZATION="TRUE" VFMX_DEV_MODE="FALSE"

- Installing Thinfinity VirtualUI Development environment only:

  c:\Thinfinity_VirtualUI_v2_Setup_x64.exe /s /v"/qn VFMX_SERVER_MODE="FALSE"

- Installing Thinfinity VirtualUI in Load Balancing mode using Gateway and Server roles and including licensing:

  C:\Thinfinity_VirtualUI_v2_Setup_x64.exe /s /v"/qn EMAIL="myemail@mydomain.com" SERIAL="45EI-09IJ-0OA6-3GH5-IT5E-E4ER-KIP9-A669" STDLBMODE="LBMODE" SM_TYPE="SM_Complete""
6 Web Interface

The web interface is composed by files that provide an environment for accessing the VirtualUI applications. It is located in the installation directory, under the 'Web' folder.

Read on to find out how to take advantage of some of the web interface components:

- login.html
- index.html
- app.html
- lab.html
- Error and Misc. Pages
6.1 login.html

The HTML login is the page where the user completes the credentials necessary to access applications that require authentication. It can either redirect the user to a specific application or to the index page, depending on how VirtualUI is configured. If only one application matches the credentials, then VirtualUI will access that application; if more than one application profile is enabled for the authenticated user, they will show as icons.

Note: This page can only be shown if standard browser authentication is not enabled.

The login.html page is distributed with the installation in the 'web' folder. This page can be modified by the developer to change the layout according to their needs.

Read more:
- index.html
- app.html
- lab.html
- Error and Misc. Pages
6.2 index.html

The index page is the page you will see by default after authenticating, except in those cases where an application is set to be accessed by default.

It shows the applications available according to the security scheme being used:
- If the end user didn't authenticate and provided that anonymous access is enabled in the 'Authentication' tab, VirtualUI will only show those applications with anonymous access.
- If the user authenticates, VirtualUI will show the anonymous applications and those that match the AD objects provided in the authentication.

- login.html
- app.html
- lab.html
- Error and Misc. Pages
6.3  app.html

This is the one of the most important web components. It’s where the remotely accessed applications are shown.

Read more:
- app.js
- The Web Toolbar
- Printing Preview and Printing
- login.html
- index.html
- lab.html
- Error and Misc. Pages
6.3.1 app.js

App.js is a javascript file distributed in the Thinfinity VirtualUI installation. Its purpose is to allow the user to modify some of the application web behaviour.

The file contains the code needed to interact with the VirtualUI objects:
var virtualUI = null;
var splash = null;

$(document).ready(function () {
    splash = new Splash();
    virtualUI = new Thinfinity.VirtualUI();
    virtualUI.onError = function (errorMsg) {
        if (errorMsg == null) {
            errorMsg = "";
        }
        splash.show("Application load failed", errorMsg, false, false);
    };
    virtualUI.onLoading = function () {
        if (virtualUI.devMode) {
            if (virtualUI.devMode) {
                splash.show("Waiting for application ...", ",", true, false);
            } else {  
                splash.show("Loading...", ",", true, false);
            }
        };
    virtualUI.onShow = function () {
        splash.hide();
    };
    virtualUI.onClose = function (url) {
        // -- The url argument is used to enable going back to a specific page
        if ((typeof url != 'undefined') && (url != '') && (url != null)) {
            // tries to close the window.top
            if ((virtualUI.devMode != true) || (window.top.opener)) {
                window.top.close();
            }
            window.top.location.href = url;
            return;
        } // if in devMode, reload the page
        if (virtualUI.devMode) {
            location.reload();
        }
    // tries to close the window/tab
        if ((virtualUI.devMode != true) || (window.opener)) {
            window.close();
        }
    // else returns to the calling page
        if ((window.top == window) && (document.referrer) && (location.href != document.referrer)) {
            location.href = document.referrer;
        } // else shows the splash
        else {
            splash.show("Application closed", ",", false, false);
        }
    });
    // Remove or comment this event to prevent onUnload message prompt.
    virtualUI.onBeforeUnload = function () {

return "You are about to close the application.";
}
// -- Connect to server...
// -- "ClientSettings" values are used during the start of the connection
// -- Set values or leave as default.
virtualUI.clientSettings = {
  // -- Leave as default, server rules (Is not necessary to assign value
  // 'CursorVisible': null,
  // 'MouseMoveGestureStyle': null,
  // 'MouseMoveGestureAction': null,
  // -- Set default values, JS rules.
  'UseViewportSize': false,
  'DockMenu': {
    // 'Enabled': true,
    'Pinned': true,
    'Items': {
      'WindowList': true,
      'ErrorReporting': true,
      'BrowserCapabilities': true,
      'Keyboard': true,
      'Fullscreen': true
    }
  }
};
// -- Or
///virtualUI.clientSettings.CursorVisible = null;
///virtualUI.clientSettings.MouseMoveGestureStyle = null;
///virtualUI.clientSettings.MouseMoveGestureAction = null;
/// -- Set default values, JS rules.
//virtualUI.clientSettings.DockMenu.Enabled = false;
virtualUI.connect();
}

Handling VirtualUI Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onError</td>
<td>This event is triggered when an error is produced in the communication or inside the VirtualUI object.</td>
</tr>
<tr>
<td>onLoading</td>
<td>This event is triggered when the page is loaded, before the onShow event.</td>
</tr>
<tr>
<td>onShow</td>
<td>This event is triggered when information from the published application starts to arrive.</td>
</tr>
<tr>
<td>onClose</td>
<td>This event is triggered when the application is closed.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>onBeforeUnload</td>
<td>Returns the message created in the new public event of the VirtualUI class. onBeforeUnload Remove or comment this event to prevent onUnload message prompt.</td>
</tr>
<tr>
<td>connect</td>
<td>This method starts the communication against the VirtualUI server.</td>
</tr>
</tbody>
</table>
The DockMenu Widget

When accessing an application through the web interface of VirtualUI, a toolbar can be found by moving the mouse (or tapping, for mobile devices) near the bottom of the screen.

The DockMenu Widget presents the following buttons:

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows list</td>
<td>This button shows the available application windows. Use it to switch between different forms of the application. For example, if a form is not accessible because it ended up below a bigger one.</td>
</tr>
<tr>
<td>Error reporting</td>
<td>Press this button to send us an error report. You can edit the subject, descriptions, and details. The 'Details' field is automatically populated with browser information that will help us understand any potential problem. This information does not include any Personally Identifiable Information (PII).</td>
</tr>
<tr>
<td>Dock</td>
<td>Use this button to dock or undock the web toolbar to the bottom of the screen. Dock it so that it doesn't disappear when you move away from it. Undock it if you don't want it to show all the time.</td>
</tr>
<tr>
<td>Browser Capabilities Info</td>
<td>Displays information about the current browser capabilities.</td>
</tr>
</tbody>
</table>

The mobile version of this toolbar also includes these additional options:

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Screen</td>
<td>This button toggles the full screen mode in mobile devices where this is supported.</td>
</tr>
<tr>
<td>Open Keyboard</td>
<td>Force the mobile keyboard to open.</td>
</tr>
</tbody>
</table>

© 2018, Cybele Software, Inc.
### 6.3.3 Printing Preview and Printing

Thinfinity VirtualUI includes a printing method that allows you to customize the printing preview dialog.

When printing remotely with a standard dialog, the application has access to the Server’s machine. In order to print with the local printers, Thinfinity VirtualUI can now send the document from the server to the browser as a pdf.

When printing, choose 'Thinfinity VirtualUI' as a printer, or create a button in your application that calls this method directly.

You will be presented with the following dialog.

<table>
<thead>
<tr>
<th>Bypass this dialog (you can change this option later)</th>
<th>Check this option to bypass this dialog and send the document directly to Direct Print or Google Cloud Print.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Cloud Print</td>
<td>Prints this document using your Google Cloud printers. If you are logged into a Google account you will see a list of available printers.</td>
</tr>
<tr>
<td>Direct Print</td>
<td>Print this document to the local printers.</td>
</tr>
</tbody>
</table>

© 2018, Cybele Software, Inc.
Ok  Sends the document to the configured printer. When pressing OK you will either see the Google Printers or the local printers.

Every time you use this printing method, you will see the following message in the top-right corner of the browser:

Preparing document..., please wait.

Bypass preview dialog OFF

This dialog lasts for a couple of seconds before you see the next printing dialog, which can be the VirtualUI preview, or the list of Google Cloud or Local printers.

If you have opted to bypass the VirtualUI preview dialog, you can access it again by clicking on the 'Preparing document...' message, so you can change whether to use Google Cloud Print or Direct Print, or not bypass the dialog the next time.

If you have opted to bypass the VirtualUI preview dialog and you press 'Escape' during the 'Preparing document...' message, you will see either the Google Cloud printers or the local printers.

If you haven't opted to bypass the preview dialog, clicking this button or pressing 'Escape' will take you to the preview dialog.
6.4 lab.html

This file is used in development mode, when the application is executed from the IDE. When you run the app in debug mode, VirtualUI sends the lab.html page to the browser instead of the app.html page.

Read more:
- The Development Lab.
- login.html
- index.html
- app.html
- Error and Misc. Pages
6.5 Error and Misc. Pages

Read about some of the other pages that make up Thinfinity VirtualUI's behaviour and how you can take advantage of them.

unsupported.html

This page shows an error message when the browser is not supported by Thinfinity VirtualUI. That is, when the browser doesn't support HTML5. You can customize the error message or even the error handling behavior by editing this page.

Error Pages

The Thinfinity VirtualUI Error Pages show up in the event of 401, 402, 403, 404, 409, or 500 HTTP responses. These pages can be edited so that you can tailor how you show each of these errors. You can find them in the 'web' folder of the installation directory, and you can also access them through the 'General' tab of the Thinfinity VirtualUI Manager.

Read more:
- login.html
- index.html
- app.html
- lab.html
7 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

7.1 Development Server

The 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 the 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 'Licenses' Tab
In the Thinfinity VirtualUI Development Server Manager 'General' tab 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
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.

<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>
7.1.2 License Manager

The license manager option is found in the License tab of Thinfinity VirtualUI Development Server Manager. Use this manager to check your licensing status, activity, add or remove your licenses.

Read more:
- The 'Licenses' Tab
- License Activation
This is how the License Manager should look once your license is registered:

- **Select**: If you registered several serials on this server, press this button to select the key you wish to use.
- **Add**: Press this button to enter your license information.
- **Remove**: Press this button if you wish to deactivate the license on this machine. This will allow you to use the license somewhere else, or to re use the license after reinstalling Windows.
- **Close**: Press this button to close the License Manager.
- **Activity**: Here you can verify in real time the amount
of users consuming a license.

Pressing the 'Add' button will open the Product Registration Wizard:

- **Get a new Trial Serial Number**: Select this option to receive a 30 day trial serial.
- **Activate a Serial Number Online**: Select this option to register your Thinfinity® VirtualUI serial.
- **Activate a Serial Number Offline**: Select this option to register a license offline.

**Read More:**
- Proxy Activation
- Get a new Trial Serial Number
- Activate a Serial Number Online
- Activate a Serial Number Offline
7.1.2.1.1 Proxy Activation

In order to register your license behind a proxy server you must register it using the Licensing Server administrator, for more information please contact support@cybelesoft.com.

Read More:
- Get a new Trial Serial Number
- Activate a Serial Number Online
- Activate a Serial Number Offline

7.1.2.1.2 Get a new Trial Serial Number

This option will allow you to request a 30 day trial license with unlimited access. You will be prompted to enter a valid name and e-mail address.

Once you filled this information hit 'Next' and check your in-box for the serial key.

Read More:
- Proxy Activation
- Activate a Serial Number Online
- Activate a Serial Number Offline
7.1.2.1.3 Activate a Serial Number Online

This is how the "Activate a Serial Number Online" windows looks:

![Activate Serial Number Window](image)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-mail</td>
<td>Enter the e-mail address you've registered with.</td>
</tr>
<tr>
<td>Serial</td>
<td>Enter the serial information we provided you.</td>
</tr>
<tr>
<td>Licensing Server URL</td>
<td>If you installed the License Server administrator, enter the License Server URL. Otherwise leave this blank.</td>
</tr>
</tbody>
</table>

If the license information is incorrect, you will see this warning: "The license information is invalid". In this case, please verify the following:
- That you are entering the exact email and Serial number sent to you. The best practice to do this correctly is to copy - paste it, being careful not to include any space after or before.
- That you have a working internet connection. If you intend to install it in a machine...
with no internet connection, you can try the Manual Activation. If you have internet restrictions because of a proxy, try the Proxy Activation.

If you need additional help, contact us.

If the license information is correct, the License Manager will let you know that "The new license has been installed successfully" and its information will be show in the License Manager.

Read More:
- Proxy Activation
- Get a new Trial Serial Number
- Activate a Serial Number Offline

7.1.2.1.4 Activate a Serial Number Offline

Manual Activation is an activation option only for those cases when you want to activate Thinfinity® VirtualUI in a machine that has no internet connection, or an internet connection restricted by heavy security policies that block a regular activation.
- If you haven't tried a regular activation, follow these instructions: Activate a Serial Number Online.
- If your internet restrictions are caused by a proxy, follow these instructions: Proxy Activation.

Before you continue with the steps to perform a manual activation, please contact us.
Once you've selected Activate a Serial Number Offline. You will see the following pop up:

![Product Registration Wizard](image)

<table>
<thead>
<tr>
<th>Serial</th>
<th>Enter the license Serial number to generate the manual activation key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate Manual Key</td>
<td>After you have entered the serial number, press this button to generate the Manual Activation Key.</td>
</tr>
<tr>
<td>Manual Activation Key</td>
<td>After you press the 'Generate Manual Key' button, a Manual Activation Key will appear in this field. Send this Manual Activation Key to support.</td>
</tr>
</tbody>
</table>
7.2 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:**

<table>
<thead>
<tr>
<th>File</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Save</td>
<td></td>
</tr>
<tr>
<td>Exit</td>
<td></td>
</tr>
</tbody>
</table>

The File Menu is composed of the following options:

| Save   | Click to save any change done on the system Settings. |
Exit
Click on this option to exit Thinfinity VirtualUI Server Manager.

Help Menu:

<table>
<thead>
<tr>
<th>Help</th>
<th>Takes you to the application online guide.</th>
</tr>
</thead>
<tbody>
<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>

Show Log:
The 'Show log' button in the bottom of the Thinfinity VirtualUI manager will open a file where the server activities, such as connecting and disconnecting, are logged.

Read more:
- The 'General' Tab
- The 'Gateways' tab
- The 'Sessions' tab
- The 'Applications' tab
- The 'Licences' tab
7.2.1 General

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

<p>| 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. |</p>
<table>
<thead>
<tr>
<th>Protocol</th>
<th>Choose between the HTTP and HTTPS protocol.</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Alert Icon]</td>
<td>Press this button to configure HTTP error responses.</td>
</tr>
<tr>
<td>![Certificate 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 certificate with your own.</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>
<tr>
<td>------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<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](#)
7.2.1.1 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>

**Status Code**
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 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](image)

| Status Code | Enter the Status Code number that you want to configure. |
| Response Action | Choose whether Thinfinity VirtualUI will show a page that is stored locally or will redirect the user to another web page. |
| Insert Content from file into the error response | 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.

**Response with a 302 redirect**

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.

Press OK to save the changes.

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

### 7.2.1.2 Managing the SSL Certificate

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.

Access the SSL certificates configuration by pressing this button:

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

**Managing the SSL Certificate**

A valid SSL certificate is included with the Thinfinity VirtualUI Server installation.
This allows you to encrypt all communications with the product's default certificate. However, browsers will typically show a security warning. Your communications are encrypted, but the browser notices the name on the certificate is not your company's. You may want to create your own certificate to identify your company and avoid this.

1. There are two ways of creating your own SSL certificate:
   a. Creating a self-signed certificate
   b. Using 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 an 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.pem).

Read more:
- The Default Embedded Certificate
- A Self-Signed Certificate
- A CA Certificate
- HTTPS Security Settings
- The 'Gateways' Tab
7.2.1.2.1 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:

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
7.2.1.2.2 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:

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

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

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

![HTTPS Security Settings 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:

![HTTPS - Security Settings dialog](image)

<table>
<thead>
<tr>
<th>Encryption Methods</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept SSL 2.0</td>
<td></td>
</tr>
<tr>
<td>Accept SSL 3.0</td>
<td></td>
</tr>
<tr>
<td>Accept TLS 1.0</td>
<td></td>
</tr>
<tr>
<td>Accept TLS 1.1</td>
<td></td>
</tr>
<tr>
<td>Accept TLS 1.2</td>
<td></td>
</tr>
</tbody>
</table>

**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

### 7.2.2 Gateways

In the Thinfinity VirtualUI Manager 'Gateways' tab you will find the following options:
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.

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 a new gateway to the Gateway List. Only if you will use Scaling and Load Balancing.
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](#)

### 7.2.3 Sessions

In the Thinfinity VirtualUI Manager 'Sessions' tab you will find the following options:

**Standard Mode Installation:**

-Thinity® VirtualUI™ requires at least one interactive Windows session. By default it uses the console session, sharing this session among all connected users.

You can configure Thinfinity® VirtualUI™ to run under an alternate Windows session or, if you installed the Gateway, you can choose to balance memory usage/performance by configuring one session per user or distribute users evenly among a number of Windows sessions.

- **Run under this account:**

  - **Username:** VirtualUI
  - **Password:** ********

- **Test**
<table>
<thead>
<tr>
<th>Run under this account</th>
<th>Check this option to enable Thinfinity VirtualUI to run applications under a separate Remote Desktop Services session.</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>Enter the username for the Remote Desktop Services session you want Thinfinity VirtualUI to run applications under.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the password for the Remote Desktop Services session you want Thinfinity VirtualUI to run applications under.</td>
</tr>
<tr>
<td>Test</td>
<td>Test the credentials entered to verify that the username and password are correct and can access RDS.</td>
</tr>
</tbody>
</table>

**Load Balancing Mode Installation:**

Thinfinity® VirtualUI™ requires at least one interactive Windows session. By default, it uses the console session, sharing this session among all connected users.

You can configure Thinfinity® VirtualUI™ to run under an alternate Windows session or, if you installed the Gateway, you can choose to balance memory usage/performance by configuring one session per user or distribute users evenly among a number of Windows sessions.

- Run under this account:
  - Username: VirtualUI
  - Password: ********
  - Session sharing: [ ]
  - Session count: 10

[Test]
<table>
<thead>
<tr>
<th>User name</th>
<th>RDS user that will be used for Load Balancing purposes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>Configures the user password</td>
</tr>
<tr>
<td>Session sharing</td>
<td>Checking this option will split all new application instances in the amount of RDS users established in the &quot;Session count&quot; field.</td>
</tr>
<tr>
<td>Session count</td>
<td>Establishes the maximum amount of RDS users that will spawn on the server.</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
7.2.4 Applications

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

<table>
<thead>
<tr>
<th>Application List</th>
<th>This list shows the available applications. You can enable or disable them by checking the box to the left of the name.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Shows the name of the application.</td>
</tr>
<tr>
<td><strong>Target</strong></td>
<td>Shows the application path or the web address for Web Link profiles.</td>
</tr>
</tbody>
</table>

**Add**
Press this button to add a new application.
<table>
<thead>
<tr>
<th><strong>Edit</strong></th>
<th>Select an application and press this button to edit it.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Remove</strong></td>
<td>Select an application and press this button to remove it.</td>
</tr>
<tr>
<td><strong>Allowed users and groups for selected profile</strong></td>
<td>See here the allowed users or group(s) of users for the selected application. If you want to change the permissions, edit the application.</td>
</tr>
<tr>
<td><strong>Database path</strong></td>
<td>Path to the profile database.</td>
</tr>
</tbody>
</table>

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

**Read more:**
- [Application Profile](#)
- [Weblink Profile](#)
- The 'Licenses' Tab
- [Scaling and Load Balancing](#)
7.2.4.1 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>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/. 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</td>
<td>Choose the landing HTML page for the application.</td>
</tr>
<tr>
<td>Page</td>
<td>Access Key</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------------</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 <strong>One-Time-URL</strong> method.</td>
</tr>
<tr>
<td>Icon</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>Applicatio n/Web link</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>Default Applicatio n</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
In the Application Profile Editor 'General' tab you will find the following options:

**Program path and file name:**
C:\AFPS\VirtualUI\EasyMoney\easyMoney.Manager.exe

**Arguments:**

**Start in the following folder:**
C:\AFPS\VirtualUI\EasyMoney\n
**Resolution:**
- Fit to browser window

**Browser rules file:**

**Reconnection timeout:** 5 seconds

---

<table>
<thead>
<tr>
<th><strong>Program path and file name</strong></th>
<th>Specify the complete path that gives access to the application executable file.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arguments</strong></td>
<td>Application arguments.</td>
</tr>
<tr>
<td><strong>Start in the following folder</strong></td>
<td>Inform a context directory for the application set on the 'Program path and file name' field.</td>
</tr>
<tr>
<td><strong>Resolution</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Browser rules file</strong></td>
<td>Specify the location of a file composed of a ruleset to adjust the remote desktop resolution according. <a href="#">Read more</a></td>
</tr>
<tr>
<td><strong>Reconnection Timeout</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

**Read more:**
- [The 'Credentials' Tab](#)
- [The 'Permissions' Tab](#)
7.2.4.1.2 Credentials

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

- **Use server's account**: Use the same credentials entered in the 'Sessions' 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 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](#)
7.2.4.1.3 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:

<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 will have access to this application. 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 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 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
7.2.4.2 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>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</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</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
## 7.2.4.2.1 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>✓ Allow anonymous access</td>
</tr>
<tr>
<td></td>
<td>Group or user names:</td>
</tr>
</tbody>
</table>

<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
7.2.5 Authentication

The 'Authentication' tab will allow you to choose the authentication methods to access VirtualUI.

Choose your authentication method(s) in the 'Method' tab:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Logon</td>
<td>Built-in</td>
</tr>
</tbody>
</table>

- Allow anonymous access
- Use standard browser authentication dialog

This list shows the available authentication methods. You can enable or disable them by checking the box to the left of the name.

- **Name**: Shows the name of the authentication method.
- **Target**: Shows the authentication method type.
<table>
<thead>
<tr>
<th><strong>Add</strong></th>
<th>Press this button to add a new authentication method. Each method presented will open a new form for you to fill in the relevant information.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edit</strong></td>
<td>Select an authentication method and press this button to edit it.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Select an authentication method and press this button to remove it.</td>
</tr>
<tr>
<td><strong>Allow anonymous access</strong></td>
<td>Check this to allow anonymous access. This means that users can access anonymous access profiles without any kind of authentication.</td>
</tr>
<tr>
<td><strong>Use standard browser authentication dialog</strong></td>
<td>Check this to use the standard browser authentication dialog. When this is unchecked, users will authenticate through the VirtualUI web login.</td>
</tr>
</tbody>
</table>

**Read more:**
- [The Mappings Tab](#)
- [OAuth2 Methods Field Reference](#)
The 'Mappings' tab of the 'Authentication' tab is where you will map all the credentials of methods other than Windows Logon to Windows Active Directory user so they can be authenticated against the profiles.

The 'Mappings' tab can show information in two different ways to ease your mapping process. By pressing the 'Switch base' button, you select whether you prefer to see a list of your authentication ID masks above, that you will map with the Associated User(s)/Group(s) Access below, or a list of Associated Permissions for Active Directory User(s) or Group(s) above that you will map to authentication IDs below. This doesn't change the way it works, only the way it is shown. You might want to think that a certain authentication method username has several Active Directory groups it's associated with and
thus choose to see the authentication method usernames above; or you might prefer to see, for example, a list of Active Directory users and link each of them with several authentication method usernames. You can try, and even go back and forth as you add users and decide which way works best for you. Switching the base doesn't change the users nor their mapping.

<table>
<thead>
<tr>
<th>Authentication ID Mask</th>
<th>This list shows your authentication ID Masks. This means that you can either use an authentication ID, or a mask that matches only some of the username's characters (the rest are represented with *).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associated Permissions</td>
<td>This list shows the Active Directory user(s) and/or group(s) associated with authentication ID masks.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Use this checkbox to enable or disable a particular authentication ID mask (only available when the Authentication ID Masks box is shown above)</td>
</tr>
<tr>
<td>Add</td>
<td>Use this button in the box above to add a new authentication ID mask or a new Active Directory user or group. Use this button in the box below after selecting an authentication ID mask, Active Directory user or group in the box above, to associate an Active Directory user or group or authentication ID mask, respectively, in the box below.</td>
</tr>
<tr>
<td>Remove</td>
<td>Use this button in the box above to remove an authentication ID mask or an Active Directory user or group. Bear in mind that this will also remove the mapping (use the 'Enabled' checkbox to disable it temporarily). Use this button in the box below to remove the mapping of an Active Directory user or group or authentication ID mask to the authentication ID mask or Active Directory user or group selected in the box above.</td>
</tr>
</tbody>
</table>
When you use RADIUS as an authentication method, you need to set some parameters:
### Authentication Method Settings

**Name:** Choose a name to identify this authentication method.

**Server IP:** Enter the RADIUS Server IP

**Port:** Enter the RADIUS Port

**Shared Secret:** Enter the RADIUS Shared Secret

**Authentication Type:**

- PAP

---

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Choose a name to identify this authentication method.</td>
</tr>
<tr>
<td>Server IP</td>
<td>Enter the RADIUS Server IP</td>
</tr>
<tr>
<td>Port</td>
<td>Enter the RADIUS Port</td>
</tr>
<tr>
<td>Shared</td>
<td>Enter the RADIUS Shared Secret</td>
</tr>
<tr>
<td>Secret</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Authentication Type</strong></td>
<td>Choose your authentication type. The 'EAP' option stands for all the EAP authentication methods.</td>
</tr>
<tr>
<td><strong>Test Configuration</strong></td>
<td>Press this button to communicate with RADIUS and test the information entered in the above fields to see if it is correct.</td>
</tr>
</tbody>
</table>
7.2.5.3 OAuth 2.0 Authentication Method Settings

When you use OAuth 2.0 as an authentication method, you need to set some parameters.

For predefined methods (Google, Facebook, LinkedIn, Dropbox), the only parameters you will need are the client ID and shared secret.
Name | Choose a name to identify this authentication method.
--- | ---
Virtual Path | Type a Virtual Path. If you access your Thinfinity VirtualUI URL followed by the virtual path:

http(s)://ip:port/virtualPath

the application will attempt to log in with this method.
If you change this value, remember to change the CSS for SSO options, setting the style for each login button. The ID for each button must match the Virtual path.

<table>
<thead>
<tr>
<th>Client ID</th>
<th>Enter your authentication provider Client ID, generated while configuring your account integration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Secret</td>
<td>Your authentication provider's Client Secret generated while configuring your account integration.</td>
</tr>
</tbody>
</table>

In the 'Server' tab of the Authentication Method Settings, you will find that the fields are completed by default for the predefined methods. Like Google in this case:
When you add an Oauth 2.0 method that is not predefined, you will need to complete these fields.

<table>
<thead>
<tr>
<th>Authorization URL</th>
<th>Enter here the URL where your authentication provider can be reached to request authorization.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization parameters</td>
<td>Additional parameters for the authorization URL</td>
</tr>
<tr>
<td><strong>Token Validation Server URL</strong></td>
<td>Enter your authentication provider's token validation server URL.</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Profile Information server URL</strong></td>
<td>Enter your authentication provider's information server URL.</td>
</tr>
<tr>
<td><strong>Login username value returned in JSON</strong></td>
<td>The name of the login username field as returned in a JSON from your authentication provider.</td>
</tr>
</tbody>
</table>
7.2.5.3.1 Configure OAuth with Auth0

This tutorial will show you how to enable 2FA using Auth0 with Thinfinity VirtualUI.

Auth0 Guardian mobile application is required for 2FA.

1) Create a new application on Auth0’s administrator site, and chose “Single Page Web Application”

Choose an application type

- **Native**
  Mobile or Desktop, apps that run natively in a device.
  - eg: iOS SDK

- **Single Page Web Applications**
  A JavaScript front-end app that uses an API.
  - eg: AngularJS + NodeJS

- **Regular Web Applications**
  Traditional web app (with refresh).
  - eg: Java ASP.NET

- **Machine to Machine Applications**
  CLI, Daemons or Services running on your backend.
  - eg: Shell Script

2) Copy your Client ID and Client Secret:
3) In the “Allowed Callback URL”, you need to add the URL that you are going to use to authenticate, and the VirtualPath of the Authentication Method (OAuth by default)

Allowed Callback URLs

https://MyThinfinityWebsite/oauth

After the user authenticates we will only call back to any of these URLs. You can specify multiple valid URLs by comma-separating them (typically to handle different environments like QA or testing). Make sure to specify the protocol (http:// or https://). Otherwise the callback may fail in some cases.

4) To enable 2FA, click on the “Multifactor Auth” and enable “Push Notifications”:
5) Open the Thinfinity VirtualUI Server manager, navigate to the authentication tab, press "Add" -> "OAuth2.0" -> "Other".
6) Add the following information:
This information can be verified in the “Endpoints” tab under Advanced Settings in the Application you created on Auth0’s interface.
Click on “OK” after you entered the information.

7) Click on the “Mappings” tab and then press “Add” under the Authentication ID Mask.
Add the email address of the Auth0 user you want to validate and press “Ok”. Then, under the “Associated Permissions” field, press on the “Add” button and search for the Active Directory User.

After you add the appropriate mappings, click on the “Apply” button.

8) Navigate to the Thinfinity’s landing page, and you should see the “Login With OAuth” option listed as an Authentication Method.
7.2.5.3.2 Configure OAuth with Okta

How to set up multifactor authentication to your environment or virtualized application.

In this quick tutorial, we will show how to properly configure Okta OAuth 2.0 for Thinfinity Remote Desktop Server and Thinfinity VirtualUI.

1) Navigate to your Okta space, go to the Applications tab, and create a new application using the “Create New App” button:
2) Select OpenID Connect as the Authentication Method:

![Create a New Application Integration](image)

3) Give the application a name, and type in the URL you use to reach Thinfinity. Then press “Save”.
4) You should be redirected to the Application Settings. In here, press the “General” button, and edit the “Login information”.

Configure the “Initiate login URI” field, by adding the Thinfinity’s website address and “/Okta” at the end of the URL.
5) Copy and past both Client ID and Client Secret for future references:

![Client Credentials](image)

6) Click on the “Assignments” tab and add your users to the Application:
7) Now, open either the Thinfinity Remote Desktop Server Manager or the Thinfinity VirtualUI Manager and navigate to the “Authentication” tab. Click on OAuth 2.0 and choose “Other”.
8) Enter your Client ID and Client Secret:
9) Click on the “Server” tab and add the following parameters:

Authorization URL: https://[MyOktaSpace].okta.com/oauth2/v1/authorize
Parameters: scope=openid+profile&state=okta
Token Validation Server URL: https://[MyOktaSpace].okta.com/oauth2/v1/token
Profile Information Server URL: https://[MyOktaSpace].okta.com/oauth2/v1/userinfo
Login username value in returned Json: preferred_username
You'll also need to change the name of the Authentication Method to “Okta” (Or to the URL you configure in the Initiate Login URI).

Press “OK” after you finish configuring the Authentication Method.

10) Click on the “Mappings” tab and then press “Add” under the Authentication ID Mask. Add the email address of the Okta user you want to validate and press “Ok”.

© 2018, Cybele Software, Inc.
Then, under the “Associated Permissions” field, press on the “Add” button and search for the Active Directory User.

After you add the appropriate mappings, click on the “Apply” button.

11) Navigate to the Thinfinity’s landing page, and you should see the “Login With Okta” option listed as an Authentication Method.
7.2.5.4 **External DLL Authentication Method Settings**

When you use your own customized external DLL as an authentication method, you only need to set the DLL.
Name

Choose a name to identify this authentication method.

External Authentication Provider

Select the DLL of your external authentication method.

Read more:
• Authentication API
7.2.5.5 Duo Authentication Method Settings

When you use Duo as an authentication method, you need to set some parameters.

| Integration Key | Enter your authentication provider Integration Key, generated while configuring your account integration. |
In the following topic we'll cover how to properly configure DUO as an authentication method using Thinfinity VirtualUI:

**How to configure DUO**

### 7.2.5.5.1 How to configure DUO

**On DUO’s Web Interface:**

1) Navigate to the Applications tab on Duo’s administrator website:

![Duo Applications Tab](image)

2) Click on "Protect an Application":
3) Create a new "Web SDK" application and click on "Protect this Application":

4) Copy the Integration Key, Secret Key, and API Hostname:
Web SDK 1

See the Duo Web SDK Documentation to integrate Duo into your custom web application.

Details

- Integration key: [Redacted]
- Secret key: Click to view. Don't write down your secret key or share it with anyone.
- API hostname: [Redacted].duosecurity.com

5) Now open the Thinfinity Remote Desktop Server Manager, navigate to the "Authentication" tab, click on "Add" and "DUO":

© 2018, Cybele Software, Inc.
6) Copy the Integration Key, Secret Key, and API Hostname provided by DUO, then click "OK" and "Apply".
7) Navigate to the Thinfinity login page, select "Use DUO" as a method of authentication, and enter valid credentials:
8) Now, you will be given the change to authenticate using a valid DUO authentication method:

Once you validate your account, you will be redirected to the index page with the Duo user validated.
7.2.5.6 SAML Authentication Method Settings

When you use Duo as an authentication method, you need to set some parameters.

![Authentication Method Settings](image)

- **Service Identifier**
- **Service Certificate File**
- **Service Certificate Password**
- **Identification Entity ID**
- **Sign Authentication Request**
- **Single Sign-On Service URL**
- **Sign-Out URL**
- **Partner Certificate File**
<table>
<thead>
<tr>
<th>Service Certificate file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Certificate Password</td>
</tr>
<tr>
<td>Identification ID</td>
</tr>
<tr>
<td>Sign Authentication Request</td>
</tr>
<tr>
<td>Single Sign/On Service URL</td>
</tr>
<tr>
<td>Sign-Out URL</td>
</tr>
<tr>
<td>Partner Certificate File</td>
</tr>
</tbody>
</table>

In the following topic we'll cover how to properly configure SAML with Okta as an authentication method using Thinfinity VirtualUI:

**Configure SAML with Okta**

7.2.5.6.1 Configure SAML with Okta

In this quick tutorial, we will show how to properly configure Okta SAML for Thinfinity Remote Desktop Server.

1) Navigate to your Okta space, go to the Applications tab, and create a new application using the "Create New App" button:
2) Chose “SAML 2.0” as the Authentication Method.

3) Assign a name to the application.
4) Configure the “Single sign-on URL” and “Audience URI”.

The “Single Sign-on URL” address should be the following: https://[MyThinfinityWebSite]/SAMLAssertionConsumerService

The Audience URI should be the URI used to connect to Thinfinity: https://
5) Choose the Feedback options that applies to your application:

6) Now that the application is created, it should redirect you to the “Settings” window. Click on “View Setup Instructions” for further information:
In here you will get the “Identity Provider Single Sign-on URL”, the Identity Provider Issuer, and the Certificate provided by Okta.

7) Now, open the Thinfinity Remote Desktop Server Manager or Thinfinity VirtualUI Server manager, navigate to the “Authentication” tab, press the “Add” option and click on “SAML”:
8) In here, you will have to add the different values provided by Okta in order to enable SAML:

- **Service Identifier = Audience URI (SP Entity ID)**
- **Service Certificate File = Your certificate’s file.**
- **Service Certificate Password = Your certificate’s password.**
- **Identificacion Entity ID = Identity Provider Issuer**
- **Single Sign-On Service URL = Identity Provider Single Sign-On URL**
- **Sign-Out URL = This value is optional.**
- **Partner Certificate File = X.509 Certificate provided by Okta.**
Below you’ll find an example on how it should look like:

After you finish adding all those values, press “Ok”.

10 ) Click on the “Mappings” tab and then press “Add” under the Authentication ID Mask.
Add the email address of the Okta user you want to validate and press “Ok”. Then, under the “Associated Permissions” field, press on the “Add” button and search for the Active Directory User.
After you add the appropriate mappings, click on the “Apply” button.

11) Navigate to the Thinfinity’s landing page, and you should see the “Login With SAML” option listed as an Authentication Method.
7.2.5.6.2 Configure SAML with Centrify

On the Centrify’s Admin Portal.

1) Click on “Apps” -> “Web Apps”:

2) Click on “Custom” and next to SAML, press “Add”
3) Give your application a name, and click on the “Trust” tab.
Click on “Manual Configuration”, and copy the IdP Entity ID, and download the certificate provided by Centrify.

4) Then copy the “Single Sign on URL”, and the “Single Logout URL”:

   ![Single Sign On URL](image1)
   ![Single Logout URL](image2)
   ![Single Sign On Error URL](image3)

5) Now, on the “Service Provider Configuration”, click on “Manual Configuration” and configure the following:

   ![Service Provider Configuration](image4)

After doing these changes, click on the “Save” button.

6) Now we need to configure Thinfinity with all this information.

Open the Server Manager and navigate to the “Authentication” tab, press “Add”, and then SAML:

7) Now we must configure the connection itself:
- Service identifier = https://YourThinfinitySite:[Port]
- Service Cert File = [Path_To_Your_Certificate]
- Service Cert Pass = [Certificate_Password]
- Identification Entity = [IdP Entity ID / Issuer]
- Single Sing on Service URL = [Single Sign on URL]
- Sign-out URL = [Single Logout URL]
- Partnet Cert File = [Certificate Provided by Centrify]

Once you configured it properly, click “Ok” and then “Apply”

8) Now go the Thinfinity landing page and you should see the “Login with SAML” option now available to use.

Sign in or select an option

```
Login with SAML
```

or

```
Sign in
```

7.2.6 License Manager

The license manager option is found in the License tab of Thinfinity VirtualUI Server Manager. Use this manager to check your licensing status, activity, add or remove your licenses.
Read more:
- License Activation

7.2.6.1 License Activation

This is how the License Manager should look once your license is registered:
Select

If you registered several serials on this server, press this button to select the key you wish to use.

Add

Press this button to enter your license information.

Remove

Press this button if you wish to deactivate the license on this machine. This will allow you to use the license somewhere else, or to re use the license after reinstalling Windows.

Close

Press this button to close the License Manager.

Activity

Here you can verify in real time the amount of users consuming a license.

Pressing the 'Add' button will open the Product Registration Wizard:
7.2.6.1.1 Proxy Activation

In order to register your license behind a proxy server you must register it using the Licensing Server administrator, for more information please contact support@cybelesoft.com.
7.2.6.1.2 Get a new Trial Serial Number

This option will allow you to request a 30 day trial license with unlimited access. You will be prompted to enter a valid name and e-mail address.

Once you filled this information hit 'Next' and check your in-box for the serial key.

Read More:
- Proxy Activation
- Activate a Serial Number Online
- Activate a Serial Number Offline

7.2.6.1.3 Activate a Serial Number Online

This is how the "Activate a Serial Number Online" window looks:
E-mail
Enter the e-mail address you've registered with.

Serial
Enter the serial information we provided you.

Licensing Server URL
If you installed the License Server administrator, enter the License Server URL. Otherwise leave this blank.

If the license information is incorrect, you will see this warning: "The license information is invalid". In this case, please verify the following:
- That you are entering the exact email and Serial number sent to you. The best practice to do this correctly is to copy - paste it, being careful not to include any space after or before.
- That you have a working internet connection. If you intend to install it in a machine with no internet connection, you can try the Manual Activation. If you have internet restrictions because of a proxy, try the Proxy Activation.

If you need additional help, contact us.

If the license information is correct, the License Manager will let you know that "The
new license has been installed successfully" and its information will be show in the License Manager.

Read More:
- Proxy Activation
- Get a new Trial Serial Number
- Activate a Serial Number Offline

7.2.6.1.4 Activate a Serial Number Offline

Manual Activation is an activation option only for those cases when you want to activate Thinfinity® VirtualUI in a machine that has no internet connection, or an internet connection restricted by heavy security policies that block a regular activation.
- If you haven't tried a regular activation, follow these instructions: Activate a Serial Number Online.
- If your internet restrictions are caused by a proxy, follow these instructions: Proxy Activation.

Before you continue with the steps to perform a manual activation, please contact us.

Once you've selected Activate a Serial Number Offline. You will see the following pop up:
Enter the Serial Number to generate an offline activation key

<table>
<thead>
<tr>
<th><strong>Serial</strong></th>
<th>Enter the license Serial number to generate the manual activation key</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generate Manual Key</strong></td>
<td>After you have entered the serial number, press this button to generate the Manual Activation Key.</td>
</tr>
<tr>
<td><strong>Manual Activation Key</strong></td>
<td>After you press the 'Generate Manual Key' button, a Manual Activation Key will appear in this field. Send this Manual Activation Key to support.</td>
</tr>
</tbody>
</table>
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.
Thinfinity® VirtualUI™

Its main menu has two sub-menus:

**File Menu:**

The File Menu is composed of the following options:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Save</strong></td>
<td>Click to save any change done on the system Settings.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Click on this option to exit Thinfinity VirtualUI Gateway Manager.</td>
</tr>
</tbody>
</table>

**Help Menu:**
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. |
| ![Warning Icon] | Press this button to configure HTTP error responses. |
| ![Certificate Icon] | 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 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 Thfinity VirtualUI log. |
8 Appendix A - Dialogs

In Thinfinity® VirtualUI™, window frames have a "web-native" interface. The same happens with standard dialog boxes, which are translated to HTML dialogs.

Find in the next topics how each one of these dialogs will be shown on the Windows and HTML5 platforms:

Message Dialogs
   - MessageDlg
   - InputBox
   - Formatted Message

Printing Dialogs
   - Page Setup
   - Print

File Dialogs
   - Open File
   - Save As
8.1 Message Dialogs

The Message Dialogs implemented in Thinfinity® VirtualUI™ are:

- MessageDlg
- InputBox
- Formatted Message
8.1.1 Message Dlg

Usage Example

```javascript
MessageDlg ('Would you like to continue?', mtConfirmation, [mbYes, mbNo], 0);
```

This is how the dialog is shown on each platform:

**Windows Platform**

![Windows Platform dialog](image1)

**HTML5 Platform**

![HTML5 Platform dialog](image2)
8.1.2 **Input Box**

**Usage Example**

```
value := InputBox( 'Input Test' , 'Please type a name' , '' );
```

This is how the dialog is shown on each platform:

**Windows Platform**

![Windows Platform Input Box]

**HTML5 Platform**

![HTML5 Platform Input Box]
8.1.3 Formatted Message

**Usage Example**

```
ShowMessageFmt( '%d/%d = %1.2f', [2, 5, 2/5] );
```

This is how the dialog is shown on each platform:

**Windows Platform**

![Windows Platform Dialog]

**HTML5 Platform**

![HTML5 Platform Dialog]
8.2 Printing Dialogs

The Printing Dialogs implemented in Thinfinity® VirtualUI™ are:

PageSetup
Print
8.2.1 Print

Usage Example

```pascal
type
  PrintDialog1: TPrintDialog;
...
PrintDialog1.Execute;
```

This is how the dialog is shown on each platform:

Windows Platform

![Windows Print Dialog]

HTML5 Platform

![HTML5 Print Dialog]
8.2.2 Page Setup

Usage Example

```pascal
type
  PageSetupDialog1: TPageSetupDialog;
...
PageSetupDialog1.Execute;
```

This is how the dialog is shown on each platform:

**Windows Platform**

![Windows Page Setup Dialog]

**HTML5 Platform**

![HTML5 Page Setup Dialog]
8.2.3 Print Preview

When printing remotely with a standard dialog, the application has access to the Server's machine. In order to print with the local printers, Thinfinity VirtualUI can now send the document from the server to the browser as a pdf.

When printing, choose 'Thinfinity VirtualUI' as a printer, or create a button in your application that calls this method directly.

You will be presented with the following dialog.

<table>
<thead>
<tr>
<th>Bypass this dialog (you can change this option later)</th>
<th>Check this option to bypass this dialog and send the document directly to Direct Print or Google Cloud Print.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Cloud Print</td>
<td>Prints this document using your Google Cloud printers. If you are logged into a Google account you will see a list of available printers.</td>
</tr>
<tr>
<td>Direct Print</td>
<td>Print this document to the local printers.</td>
</tr>
</tbody>
</table>
Ok | Sends the document to the configured printer. When pressing OK you will either see the Google Printers or the local printers.

Every time you use this printing method, you will see the following message in the top-right corner of the browser:

Preparing document..., please wait.
Bypass preview dialog OFF

This dialog lasts for a couple of seconds before you see the next printing dialog, which can be the VirtualUI preview, or the list of Google Cloud or Local printers.

- If you have opted to bypass the VirtualUI preview dialog, you can access it again by clicking on the 'Preparing document...' message, so you can change whether to use Google Cloud Print or Direct Print, or not bypass the dialog the next time.

- If you have opted to bypass the VirtualUI preview dialog and you press 'Escape' during the 'Preparing document...' message, you will see either the Google Cloud printers or the local printers.

- If you haven't opted to bypass the preview dialog, clicking this button or pressing 'Escape' will take you to the preview dialog.

8.3 File Dialogs

For these dialogs, you can use the standard dialogs or our web-native alternative.

The standard dialog is handled by Windows in the computer where Thinfinity VirtualUI Server is installed. The web dialog runs in the client machine. This can make a difference, for example, when opening a file: a standard dialog will show the server's files and folder, while the web dialog will show the client file system.

Set this option using the StdDialogs property, set it to 'true' to use the standard dialog, and set it to 'false' to use Thinfinity VirtualUI web dialogs. Check the Symbol reference for the programming language you have chosen.

The File Dialogs implemented in Thinfinity® VirtualUI™ are:

Open File
Save As
8.3.1 Open File

Usage Example

```pascal
type
  OpenFileDialog1: TOpenDialog;
...

OpenDialog1.Title := 'Open file';
OpenDialog1.Execute;
```

This is how the dialog is shown on each platform:

**Windows Platform**

![Windows Platform Dialog]

**HTML5 Platform**

Standard Dialog:
Non Standard Dialog:

In order to open a File from the remote machine the user will be able to make an upload of one local file into the server.
8.3.2 Save As

Usage Example

```pascal
type
    SaveDialog1: TSaveDialog;
...

SaveDialog1.Title := 'Save As...';
SaveDialog1.Filter := '*.txt';
SaveDialog1.FileName := 'somefile.txt';
SaveDialog1.Execute;
```

This is how the dialog is shown on each platform:

**Windows Platform**

![Windows Save Dialog]

**HTML5 Platform**

**Standard Dialog:**
Non Standard Dialog:

In order to save a File from the remote machine Thinfinity® VirtualUI™ will do a download into the LocalMachine.
9 Appendix B - Tailoring the interface

9.1 Customizing the Web Interface

Thinfinity® VirtualUI™ allows you to modify the web interface and tailor it to your branding scheme.

Customizing the application logo and other image files can be very simple, once it only requires you to have the new image file and tell the application where it is located.

Customizing the structure and style of the application may be a little bit more complex. These kind of customizations have to be done at a programming level (HTML and CSS).

Read also how to protect the customized web files in the Files Location topic.
9.1.1 Changing the Logo

Modifying the application logo can be as simple as copying the new logo image and telling Thinfinity® VirtualUI™ application where it is located:

1. Create a folder called "BrandingFiles", if it doesn't exist yet, under the folder web located inside the Thinfinity® VirtualUI™ installation directory. (e.g.: C:/Program Files/Thinfinity® VirtualUI™/web)

2. Copy your own logo image file to the "BrandingFiles" folder.

3. Create the WebAliases.ini file and configure it:
   a. Create a file called "WebAliases.ini" in the installation directory (e.g.: C:/Program Files/Thinfinity® VirtualUI™/bin32/WebAliases.ini or C:/Program Files/Thinfinity® VirtualUI™/bin64/WebAliases.ini, depending on your version). If the file already exists, only append the lines to it.
   b. Configure the redirection of the logo files you want to substitute, following the two examples below (Thinfinity_Virtual_UI_150.png.png and favicon.ico):

   ```
   [Alias]
   ;=================
   ;Main logo
   ;=================
   /images/Thinfinity_Virtual_UI_150.png.png=BrandingFiles\MyLogo.png
   
   ;=================
   ;Favicon
   ;=================
   /favicon.ico=BrandingFiles\MyFavicon.ico
   ```
   c. Save it.

4. Open the application to see the changes.

Bear in Mind:

a. Any line in the "WebAliases.ini" file starting with a semicolon will not be considered by the application. It can be used to leave comments in the file.

b. You can substitute any interface image or file, by following the same steps described above.

c. Sometimes the favicon is not shown right the way, because the browser keeps history of the images. In that case, you should clean the browser cache before trying out the changes.
9.1.2 Customizing the Web Files

To customize the web files, you should:

1. Create a folder called "BrandingFiles", if it doesn't exist yet, under the folder web located inside the Thinfinity® VirtualUI™ installation directory. (e.g.: C:/Program Files/Thinfinity® VirtualUI™/web)

2. Make copies of the original web files that you want to modify to the "BrandingFiles" folder. Copy only the files to be modified without their associated folder structure.

3. Customize the files (html, css, etc) as you prefer.

4. Create the WebAliases.ini file and configure it:

   a. Create a file called "WebAliases.ini" in the installation directory (e.g.: C:/Program Files/Thinfinity® VirtualUI™/bin32/WebAliases.ini or C:/Program Files/Thfinity® VirtualUI™/bin64/WebAliases.ini, depending on your version). If the file already exists, only append the lines to it.

   b. Configure the redirection to the files you have modified, by adding a line similar to the examples below for each modified file:

   ```
   [Alias]
   /index.html=BrandingFiles\my_index.html
   /css/index.css=BrandingFiles\my_index.css
   ```

   c. Save it.

5. Open the application and check out the changes.

Bear in Mind:

a. Any line in the "WebAliases.ini" file that starts with a semicolon will not be considered by the application. It can be used to leave comments.

b. The paths located in the HTML, CSS, and other contents will be kept relative to the original file location. This means that you won't have to change the content paths when customizing this files.
9.1.3 Files Location

We recommend that a new folder be created in order to keep the customized files instead of having them with the original files. This will enable you to:

a) Get back to the original interface configuration at any time.
b) Make sure that your files will be safe after a version upgrade.

You can also choose to place the files inside or outside the webroot structure. Keep reading to see how each option will behave:

**Storing the Customized Files in the Webroot Directory:**

In this case:

1) The files will be externally accessible from a URL similar to: https://127.0.0.1/BrandingFiles/customizedFile.html

2) The file paths, indicated in the "WebAliases.ini", can be relative to the webroot directory. (e.g. "/img/Thinfinity_Virtual_UI_150.png.png=BrandingFiles/MyLogo.png"). You will find other relative path examples on the topics [Changing the logo](#) and [Customizing the web files](#).

**Storing the Customized Files Outside the Webroot Directory:**

In that case:

1) The files will be protected, because it won’t be possible to access the customized files from a URL.
2) The file paths, indicated in the "WebAliases.ini", must be absolute, as in the example below:

```ini
[Alias]
/index.html=c:/BrandingFiles/my_index.html
/images/Thinfinity_Virtual_UI_150.png=c:/BrandingFiles/MyLogo.png
```
10 Symbol Reference

10.1 ActiveX Interfaces

Libraries

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualUIS</td>
<td>Thinfinity VirtualUI settings library</td>
</tr>
<tr>
<td>Thinfinity</td>
<td>Thinfinity VirtualUI library</td>
</tr>
</tbody>
</table>

10.1.1 Server Configuration Library

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>This is class VirtualUIS::Server.</td>
</tr>
<tr>
<td>Gateways</td>
<td>This is class VirtualUIS::Gateways.</td>
</tr>
</tbody>
</table>

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>
<tr>
<td>Gateways</td>
<td>This is class VirtualUIS::Gateways.</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>IProfiles</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>
</tbody>
</table>
ICertificate
Manages the certificate's configuration for HTTPS Binding.

IRDS
Manages the configuration of a Remote Desktop Services account.

IRDSAccounts
Contains a list of Remote Desktop Services accounts.

IGateways
This is class VirtualUIS::IGateways.

### Topics
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>!!OVERLOADED_ID_VirtualUIS::IProfile</td>
<td>Internal ID of the profile. This value is auto generated by the library when the profile is created.</td>
</tr>
<tr>
<td>!!OVERLOADED_Name_VirtualUIS::IProfile</td>
<td>Profile name. Is the caption for the Application or Web link shown in VirtualUI's home page.</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>

#### 10.1.1.1 IServer Interface

Gives you access to server's settings

#### Class Hierarchy

```
[ uuid(845B4EE8-0F67-4D84-A4CE-642BBD520A47), helpstring("Interface for VirtualUIConfig Object"), dual, oleautomation ]
interface IServer : IDispatch;
```

#### File

VirtualUIS.ridl

#### Library

Server Configuration Library

#### 10.1.1.1.1 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>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>
</tbody>
</table>

### 10.1.1.1.1 IServer::Load Method

Loads all the configuration entries and profiles from file. It is automatically called by constructor.

**IDL**

```idl
[id(0x000000CD)]
HRESULT _stdcall Load();
```

**Group**

**IServer Methods**

### 10.1.1.1.2 IServer::Save Method

Saves the entire configuration parameters and profiles.

**IDL**

```idl
[id(0x000000CE)]
HRESULT _stdcall Save();
```

**Group**

**IServer Methods**

### 10.1.1.1.3 IServer::HideSection Method

Hides a configuration section in the VirtualUI Server Manager.

**IDL**

```idl
[id(0x000000CF)]
HRESULT _stdcall HideSection([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.  
  * SRVSECAPPLICATIONS: Hides the list of applications.  |
**Group**

**IServer Methods**

### 10.1.1.1.4 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. |

**Group**

**IServer Methods**

### 10.1.1.1.2 IServer Properties

The properties of the IServer class are listed here.

**Interface**

**IServer Interface**

**Public Properties**

<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.</td>
</tr>
</tbody>
</table>
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.

| Profiles       | Returns the profiles list. |
| License        | Returns the current VirtualUI Server's licence. |
| Gateways       | This is Gateways, a member of class IServer. |

### 10.1.1.1.2.1 IServer::Binding Property

Returns the TCP/IP binding parameters

```idl
__property IBinding** Binding;
```

See Also

IBinding interface

Group

IServer Properties

### 10.1.1.2.2 IServer::Certificate Property

Returns the SSL certificate parameters

```idl
__property ICertificate** Certificate;
```

See Also

ICertificate interface

Group

IServer Properties

### 10.1.1.2.3 IServer::RDSAccounts Property

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
```

© 2018, Cybele Software, Inc.
_property IRDSAccounts** RDSAccounts;

See Also
IRDSAccounts interface

Group
IServer Properties

10.1.1.1.2.4 IServer::Profiles Property

Returns the profiles list.

IDL
__property IProfiles** Profiles;

See Also
IProfiles interface

Group
IServer Properties

10.1.1.2.5 IServer::License Property

Returns the current VirtualUI Server's licence.

IDL
__property ILicense** License;

See Also
ILicense interface

Group
IServer Properties

10.1.1.2.6 IServer::Gateways Property

IDL
__property IGateways** Gateways;

Description
This is Gateways, a member of class IServer.
10.1.1.2 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

10.1.1.2.1 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>

10.1.1.2.1.1 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>
**Returns**

True if the license was successfully activated. False otherwise (in which case check resultCode and resultText).

**Group**

ILicense Methods

10.1.1.2.1.2 ILicense::Deactivate Method

Deactivates a previously activated license.

**IDL**

```idl
[id(0x000000CF)]
HRESULT __stdcall Deactivate();
```

**Group**

ILicense Methods

10.1.1.2.2 Properties

10.1.1.2.2.1 CustomerID

Customer identification.

**Interface**

ILicense Interface

10.1.1.2.2.2 Limits

Returns the License limits, if any (ie, trial days, max servers, max users per installation, etc).

**Interface**

ILicense Interface

10.1.1.2.2.3 Features

Returns custom features enabled on the License, if any.
### Interface

**ILicense Interface**

#### 10.1.1.2.4 IsTrial

Returns true if the current License is in trial mode.

### Interface

**ILicense Interface**

#### 10.1.1.2.5 SerialStr

Serial number of the current License.

### Interface

**ILicense Interface**

#### 10.1.1.2.3 ILicense Properties

The properties of the ILicense class are listed here.

### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerID</td>
<td>This is CustomerID, a member of class ILicense.</td>
</tr>
<tr>
<td>IsTrial</td>
<td>This is IsTrial, a member of class ILicense.</td>
</tr>
<tr>
<td>SerialStr</td>
<td>This is SerialStr, a member of class ILicense.</td>
</tr>
</tbody>
</table>

#### 10.1.2.3.1 ILicense::CustomerID Property

**IDL**

```idl
__property BSTR CustomerID;
```

**Description**

This is CustomerID, a member of class ILicense.

**Group**
ILicense Properties

10.1.1.2.3.2 ILicense::IsTrial Property

IDL

__property VARIANT_BOOL IsTrial;

Description

This is IsTrial, a member of class ILicense.

Group

ILicense Properties

10.1.1.2.3.3 ILicense::SerialStr Property

IDL

__property BSTR SerialStr;

Description

This is SerialStr, a member of class ILicense.

Group

ILicense Properties

10.1.1.3 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
## 10.1.1.3.1 Properties

### Interface

**IProfile Interface**

### 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>

### 10.1.1.3.1.1 ID

Internal profile ID. This value is auto generated by the library when the profile is created.
10.1.1.3.1.2 Name

Profile name. It's the caption shown under the application's icon in VirtualUI index page.

10.1.1.3.1.3 VirtualPath

The Virtual Path unique to this profile.

10.1.1.3.1.4 IsDefault

This option is used to make this profile the default application.

10.1.1.3.1.5 Enabled

Enables or disables the profile. Disabled profiles are not accessible by users.

10.1.1.3.1.6 ProfileKind

Gets or sets the profile type: Application or Web Link. Uses the PROFILE_APP and PROFILE_WEBLINK constants.
10.1.1.3.1.7 FileName

Complete path of the application executable file. Only used when the ProfileKind is Application.

❖ Group

Properties

10.1.1.3.1.8 Arguments

Parameters to be passed to application.

❖ Group

Properties

10.1.1.3.1.9 StartDir

Application startup directory. In most cases, the same directory of the application executable file.

❖ Group

Properties

10.1.1.3.1.10 UserName

A valid Windows User account to run the application.

❖ Group

Properties

10.1.1.3.1.11 Password

Password of the Windows User account.

❖ Group

Properties

10.1.1.3.1.12 ScreenResolution

Screen resolution in the browser. Uses the constants SCREENRES_...
10.1.1.3.1.13 WebLink

Complete Web Link URL (used only when ProfileKind is Web Link).

- **Group**
  - **Properties**

10.1.1.3.1.14 HomePage

Use it to set a customized home page for the application.

- **Group**
  - **Properties**

10.1.1.3.1.15 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**

10.1.1.3.1.16 IconData

Contains the icon of the profile, consisting in a PNG image in base64 format.

- **Group**
  - **Properties**

10.1.1.3.2 IProfile Properties

The properties of the IProfile class are listed here.

- **Interface**
  - **IProfile Interface**

- **Public Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>This is ID, a member of class IProfile.</td>
</tr>
<tr>
<td>Name</td>
<td>This is Name, a member of class IProfile.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VirtualPath</td>
<td>This is VirtualPath, a member of class IProfile.</td>
</tr>
<tr>
<td>IsDefault</td>
<td>This is IsDefault, a member of class IProfile.</td>
</tr>
<tr>
<td>Enabled</td>
<td>This is Enabled, a member of class IProfile.</td>
</tr>
<tr>
<td>ProfileKind</td>
<td>This is ProfileKind, a member of class IProfile.</td>
</tr>
<tr>
<td>FileName</td>
<td>This is FileName, a member of class IProfile.</td>
</tr>
<tr>
<td>Arguments</td>
<td>This is Arguments, a member of class IProfile.</td>
</tr>
<tr>
<td>StartDir</td>
<td>This is StartDir, a member of class IProfile.</td>
</tr>
<tr>
<td>UserName</td>
<td>This is UserName, a member of class IProfile.</td>
</tr>
<tr>
<td>Password</td>
<td>This is Password, a member of class IProfile.</td>
</tr>
<tr>
<td>ScreenResolution</td>
<td>This is ScreenResolution, a member of class IProfile.</td>
</tr>
<tr>
<td>WebLink</td>
<td>This is WebLink, a member of class IProfile.</td>
</tr>
<tr>
<td>HomePage</td>
<td>This is HomePage, a member of class IProfile.</td>
</tr>
<tr>
<td>IdleTimeout</td>
<td>This is IdleTimeout, a member of class IProfile.</td>
</tr>
<tr>
<td>IconData</td>
<td>This is IconData, a member of class IProfile.</td>
</tr>
</tbody>
</table>

### 10.1.1.3.2.1 IProfile::ID Property

**IDL**

```idl
__property BSTR ID;
```

**Description**

This is ID, a member of class IProfile.

**Group**

IProfile Properties

### 10.1.1.3.2.2 IProfile::Name Property

**IDL**

```idl
__property BSTR Name;
```

**Description**

This is Name, a member of class IProfile.

**Group**

IProfile Properties
10.1.1.3.2.3  IProfile::VirtualPath Property

IDL
__property BSTR VirtualPath;

- Description
This is VirtualPath, a member of class IProfile.

- Group
IProfile Properties

10.1.1.3.2.4  IProfile::IsDefault Property

IDL
__property VARIANT_BOOL IsDefault;

- Description
This is IsDefault, a member of class IProfile.

- Group
IProfile Properties

10.1.1.3.2.5  IProfile::Enabled Property

IDL
__property VARIANT_BOOL Enabled;

- Description
This is Enabled, a member of class IProfile.

- Group
IProfile Properties

10.1.1.3.2.6  IProfile::ProfileKind Property

IDL
__property enum ProfileKind ProfileKind;

- Description
This is ProfileKind, a member of class IProfile.

- Group
IProfile Properties

10.1.1.3.2.7 IProfile::FileName Property

IDL
__property BSTR FileName;

Description
This is FileName, a member of class IProfile.

Group
IProfile Properties

10.1.1.3.2.8 IProfile::Arguments Property

IDL
__property BSTR Arguments;

Description
This is Arguments, a member of class IProfile.

Group
IProfile Properties

10.1.1.3.2.9 IProfile::StartDir Property

IDL
__property BSTR StartDir;

Description
This is StartDir, a member of class IProfile.

Group
IProfile Properties

10.1.1.3.2.10 IProfile::UserName Property

IDL
__property BSTR UserName;

Description
This is UserName, a member of class IProfile.
10.1.1.3.2.11  IProfile::Password Property

IDL
__property BSTR Password;

Description
This is Password, a member of class IProfile.

10.1.1.3.2.12  IProfile::ScreenResolution Property

IDL
__property enum ScreenResolution ScreenResolution;

Description
This is ScreenResolution, a member of class IProfile.

10.1.1.3.2.13  IProfile::WebLink Property

IDL
__property BSTR WebLink;

Description
This is WebLink, a member of class IProfile.

10.1.1.3.2.14  IProfile::HomePage Property

IDL
__property BSTR HomePage;

Description
This is HomePage, a member of class IProfile.

- **Group**
  - IProfile Properties

10.1.1.3.2.15 IProfile::IdleTimeout Property

**IDL**

```idl
__property long IdleTimeout;
```

- **Description**
  This is IdleTimeout, a member of class IProfile.

- **Group**
  - IProfile Properties

10.1.1.3.2.16 IProfile:: IconData Property

**IDL**

```idl
__property BSTR IconData;
```

- **Description**
  This is IconData, a member of class IProfile.

- **Group**
  - IProfile Properties

10.1.1.4 IProfiles Interface

Contains the list of profiles registered in VirtualUI Server.

- **Class Hierarchy**

```
IDISpatch  \---- VirtualUIS::IProfiles
```

**IDL**

```
[ uuid(C271394D-82FA-4DF9-A603-9927AA76A4F9), dual, oleautomation ]
interface IProfiles : IDispatch;
```

- **File**
  VirtualUIS.ridl

- **Library**
Server Configuration Library

10.1.1.4.1 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>

10.1.1.4.1.1 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

10.1.1.4.1.2 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

10.1.1.4.2 Properties

10.1.1.4.2.1 Count

Returns the profile count.

Interface

IProfiles Interface

10.1.1.4.2.2 Item

Returns a profile from the list by its index.

Description

Profile interface.

See Also

IProfile

Interface

IProfiles Interface

10.1.1.5 IBinding Interface

Interface for the server's binding parameters.

Class Hierarchy

IDL

```
[ uuid(52C63E8D-2FA4-4179-AFDB-2D33853F3356), dual, oleautomation ]
interface IBinding : IDispatch;
```

File

VirtualUIS.ridl

Library

Server Configuration Library
10.1.1.5.1 Properties

10.1.1.5.1.1 Protocol

Gets or sets the network protocol: HTTP or HTTPS. Uses the PROTO_HTTP and PROTOHTTPS constants.

IDL

__property enum Protocol Protocol;

Description:

IBinding Properties

The properties of the IBinding class are listed here.

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>This is Protocol, a member of class IBinding.</td>
</tr>
<tr>
<td>IPAddress</td>
<td>This is IPAddress, a member of class IBinding.</td>
</tr>
<tr>
<td>Port</td>
<td>This is Port, a member of class IBinding.</td>
</tr>
</tbody>
</table>

10.1.1.5.2.1 IBinding::Protocol Property

IDL

__property enum Protocol Protocol;
This is Protocol, a member of class IBinding.

**Group**

**IBinding Properties**

### 10.1.1.5.2.2 IBinding::IPAddress Property

**IDL**

```idl
__property BSTR IPAddress;
```

**Description**

This is IPAddress, a member of class IBinding.

**Group**

**IBinding Properties**

### 10.1.1.5.2.3 IBinding::Port Property

**IDL**

```idl
__property long Port;
```

**Description**

This is Port, a member of class IBinding.

**Group**

**IBinding Properties**

### 10.1.1.6 ICertificate Interface

Manages the certificate's configuration for HTTPS Binding.

**Class Hierarchy**

```
IDispatch → VirtualUIS:ICertificate
```

**IDL**

```idl
[ uuid(8B534446-EDC5-4EE7-91B0-13B5DACC5B51), dual, oleautomation ]
interface ICertificate : IDispatch;
```

**File**

VirtualUIS.ridl

**Library**
Server Configuration Library

10.1.1.6.1 Properties

10.1.1.6.1.1 CertFile

Gets or sets the Certificate file path.

≡ Interface
ICertificate Interface

10.1.1.6.1.2 CAFile

Gets or sets the Certificate Authority file path.

≡ Interface
ICertificate Interface

10.1.1.6.1.3 PKFile

Gets or sets the path of Private Key file.

≡ Interface
ICertificate Interface

10.1.1.6.1.4 PassPhrase

Gets or sets the certificate password.

≡ Interface
ICertificate Interface

10.1.1.6.2 ICertificate Properties

The properties of the ICertificate class are listed here.

≡ Interface
ICertificate Interface

≡ Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2018, Cybele Software, Inc.
10.1.1.6.2.1 ICertificate::CertFile Property

**IDL**

```
__property BSTR CertFile;
```

**Description**

This is CertFile, a member of class ICertificate.

**Group**

ICertificate Properties

10.1.1.6.2.2 ICertificate::CAFile Property

**IDL**

```
__property BSTR CAFile;
```

**Description**

This is CAFile, a member of class ICertificate.

**Group**

ICertificate Properties

10.1.1.6.2.3 ICertificate::PKFile Property

**IDL**

```
__property BSTR PKFile;
```

**Description**

This is PKFile, a member of class ICertificate.

**Group**

ICertificate Properties
10.1.1.6.2.4 ICertificate::PassPhrase Property

**IDL**

```idl
__property BSTR PassPhrase;
```

**Description**

This is PassPhrase, a member of class ICertificate.

**Group**

ICertificate Properties

10.1.1.7 IRDS Interface

Manages the configuration of a Remote Desktop Services account.

**Class Hierarchy**

[Dispatch] ➔ [VirtualUIS:IRDS]

**IDL**

```idl
[ uuid(103B86C8-E012-4AC7-A366-D3845BBB8D5E), dual, oleautomation ]
interface IRDS : IDispatch;
```

**File**

VirtualUIS.ridl

**Library**

Server Configuration Library

10.1.1.7.1 Properties

10.1.1.7.1.1 Enabled

Enable or disable the use of this RDS account.

**Interface**

IRDS Interface

10.1.1.7.1.2 UserName

**Summary**

Gets or sets the RDS Username.

**Interface**
10.1.1.7.1.3 Password

Gets or sets the RDS Password.

10.1.1.7.2 IRDS Properties

The properties of the IRDS class are listed here.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserName</td>
<td>This is UserName, a member of class IRDS.</td>
</tr>
<tr>
<td>Password</td>
<td>This is Password, a member of class IRDS.</td>
</tr>
</tbody>
</table>

10.1.1.7.2.1 IRDS::UserName Property

IDL

```idl
__property BSTR UserName;
```

Description

This is UserName, a member of class IRDS.

Group

IRDS Properties

10.1.1.7.2.2 IRDS::Password Property

IDL

```idl
__property BSTR Password;
```

Description

This is Password, a member of class IRDS.

Group
IRDS Properties

10.1.1.8 IRDSAccounts Interface

Contains a list of Remote Desktop Services accounts.

**Class Hierarchy**

IDL
[uuid(60666BC2-7E17-4842-9716-CFA3DCFD5583), dual, oleautomation]
interface IRDSAccounts : IDispatch;

**File**
VirtualUIS.ridl

**Library**
Server Configuration Library

10.1.1.8.1 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>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>

10.1.1.8.1.1 IRDSAccounts::Item Method

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**
10.1.1.8.1.2 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

10.1.1.8.1.3 IRDSAccounts::Delete Method

**IDL**

```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

10.1.1.8.2 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>This is Count, a member of class IRDSAccounts.</td>
</tr>
</tbody>
</table>

10.1.1.8.2.1 IRDSAccounts::Count Property

**IDL**

```idl
__property long* Count;
```

**Description**
This is Count, a member of class IRDSAccounts.

œ Group
IRDSAccounts Properties

10.1.1.9 IGateways Interface

œ Class Hierarchy

IDL
[ uuid(716BBB17-7A57-46D1-93BB-2C8A947E1F6B), dual, oleautomation ]
interface IGateways : IDispatch;

œ File
VirtualUIS.ridl

œ Description
This is class VirtualUIS::IGateways.

œ Library
Server Configuration Library

10.1.1.9.1 IGateways Methods

The methods of the IGateways class are listed here.

œ Interface
IGateways Interface

œ Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>This is Item, a member of class IGateways.</td>
</tr>
<tr>
<td>Add</td>
<td>This is Add, a member of class IGateways.</td>
</tr>
<tr>
<td>Delete</td>
<td>This is Delete, a member of class IGateways.</td>
</tr>
</tbody>
</table>

10.1.1.9.1.1 IGateways::Item Method

IDL
[propget, id(0x000000CA), defaultcollelem]
HRESULT _stdcall Item([in] long index, [out, retval] BSTR* Value);
Description
This is Item, a member of class IGateways.

Group
IGateways Methods

10.1.1.9.1.2 IGateways::Add Method

IDL
[id(0x000000CB)]
HRESULT _stdcall Add([in] BSTR URL);

Description
This is Add, a member of class IGateways.

Group
IGateways Methods

10.1.1.9.1.3 IGateways::Delete Method

IDL
[id(0x000000CC)]
HRESULT _stdcall Delete([in] long Index);

Description
This is Delete, a member of class IGateways.

Group
IGateways Methods

10.1.1.9.2 IGateways Properties

The properties of the IGateways class are listed here.

Interface
IGateways Interface

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Count]</td>
<td>This is Count, a member of class IGateways.</td>
</tr>
</tbody>
</table>
10.1.1.9.2.1 IGateways::Count Property

**IDL**

```IDL
__property long* Count;
```

**Description**

This is Count, a member of class IGateways.

**Group**

IGateways Properties

10.1.2 Thinfinity Library

Thinfinity VirtualUI library

**CoClasses**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualUI</td>
<td>This is class Thinfinity::VirtualUI.</td>
</tr>
<tr>
<td>BrowserInfo</td>
<td>This is class Thinfinity::BrowserInfo.</td>
</tr>
<tr>
<td>DevServer</td>
<td>This is class Thinfinity::DevServer.</td>
</tr>
<tr>
<td>ClientSettings</td>
<td>This is class Thinfinity::ClientSettings.</td>
</tr>
<tr>
<td>JSValue</td>
<td>This is class Thinfinity::JSValue.</td>
</tr>
<tr>
<td>JSNamedValue</td>
<td>This is class Thinfinity::JSNamedValue.</td>
</tr>
<tr>
<td>JSObject</td>
<td>This is class Thinfinity::JSObject.</td>
</tr>
<tr>
<td>JSObjects</td>
<td>This is class Thinfinity::JSObjects.</td>
</tr>
<tr>
<td>JSProperty</td>
<td>This is class Thinfinity::JSProperty.</td>
</tr>
<tr>
<td>JSProperties</td>
<td>This is class Thinfinity::JSProperties.</td>
</tr>
<tr>
<td>JSArgument</td>
<td>This is class Thinfinity::JSArgument.</td>
</tr>
<tr>
<td>JSArguments</td>
<td>This is class Thinfinity::JSArguments.</td>
</tr>
<tr>
<td>JSMETHOD</td>
<td>This is class Thinfinity::JSMETHOD.</td>
</tr>
<tr>
<td>JSMETHODS</td>
<td>This is class Thinfinity::JSMETHODS.</td>
</tr>
<tr>
<td>JSEvent</td>
<td>This is class Thinfinity::JSEvent.</td>
</tr>
<tr>
<td>JSEvents</td>
<td>This is class Thinfinity::JSEvents.</td>
</tr>
<tr>
<td>Recorder</td>
<td>This is class Thinfinity::Recorder.</td>
</tr>
<tr>
<td>RecTrack</td>
<td>This is class Thinfinity::RecTrack.</td>
</tr>
<tr>
<td>RecTracks</td>
<td>This is class Thinfinity::RecTracks.</td>
</tr>
<tr>
<td>FileSystemFilter</td>
<td>This is class Thinfinity::FileSystemFilter.</td>
</tr>
<tr>
<td>RegistryFilter</td>
<td>This is class Thinfinity::RegistryFilter.</td>
</tr>
</tbody>
</table>
## Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJSDataType</td>
<td>This is record Thinfinity::IJSDataType.</td>
</tr>
<tr>
<td>Orientation</td>
<td>This is record Thinfinity::Orientation.</td>
</tr>
<tr>
<td>MouseMoveGestureStyle</td>
<td>This is record Thinfinity::MouseMoveGestureStyle.</td>
</tr>
<tr>
<td>MouseMoveGestureAction</td>
<td>This is record Thinfinity::MouseMoveGestureAction.</td>
</tr>
<tr>
<td>RecorderState</td>
<td>This is record Thinfinity::RecorderState.</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>Interface to 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>Allows for managing the Development Server.</td>
</tr>
<tr>
<td>IClientSettings</td>
<td>Controls working parameters on the client side.</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 a method called on the client side and executed on the server side.</td>
</tr>
<tr>
<td>IJSMethods</td>
<td>Collection of methods.</td>
</tr>
</tbody>
</table>
### Thinfinity::IJSDDataType 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 Thinfinity::IJSDDataType.

**Library**
Thinfinity Library
10.1.2.2 Thinfinity::Orientation Enumeration

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

File
VirtualUIX.ridl

Description
This is record Thinfinity::Orientation.

Library
Thinfinity Library

10.1.2.3 Thinfinity::MouseMoveGestureStyle Enumeration

IDL
[ uuid(B1A699F2-F16C-49AA-93DD-1A056936AA87) ]
enum MouseMoveGestureStyle {
  MM_STYLE_RELATIVE = 0,
  MM_STYLE_ABSOLUTE = 1
};

File
VirtualUIX.ridl

Description
This is record Thinfinity::MouseMoveGestureStyle.

Library
Thinfinity Library

10.1.2.4 Thinfinity::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 Thinfinity::MouseMoveGestureAction.

Library

Thinfinity Library

10.1.2.5 Thinfinity::RecorderState Enumeration

IDL

```
[ uuid(E27758AD-1EF1-45BA-A66B-7B8FEE8AF1E1) ]
enum RecorderState {
    Inactive = 0,
    Recording = 1,
    Playing = 2
};
```

File

VirtualUIX.ridl

Description

This is record Thinfinity::RecorderState.

Library

Thinfinity Library

10.1.2.6 IVirtualUI Interface

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

Class Hierarchy

```
IDispatch  Thinfinity::IVirtualUI
```

IDL

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

File

VirtualUIX.ridl
10.1.2.6.1 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>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>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>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>UploadFile</td>
<td>Selects a file from client machine, and it’s uploaded to ServerDirectory.</td>
</tr>
<tr>
<td>TakeScreenshot</td>
<td>Takes a screenshot of a Window.</td>
</tr>
<tr>
<td>PreviewPdf</td>
<td>Sends the specified PDF file to be shown on the end-user’s web browser. It’s similar to PrintPdf, except that disables the printing options in the browser. Built-in browser printing commands will be available.</td>
</tr>
</tbody>
</table>

10.1.2.6.1.1 IVirtualUI::Start Method

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

**IDL**

```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 established within the time specified by Timeout parameter.

Group

IVirtualUI Methods

10.1.2.6.1.2 IVirtualUI::Stop Method

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

IDL

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

Group

IVirtualUI Methods

10.1.2.6.1.3 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)]
```

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
10.1.2.6.1.4  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>[in] BSTR AFileName</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**

IVirtualUI Methods

10.1.2.6.1.5  IVirtualUI::OpenLinkDlg Method

Displays a popup with a button to open a web link.

**IDL**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR url</td>
<td>Link to open.</td>
</tr>
<tr>
<td>[in] BSTR caption</td>
<td>Text to display in popup.</td>
</tr>
</tbody>
</table>

**Group**

IVirtualUI Methods

10.1.2.6.1.6  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.
### 10.1.2.6.1.7 IVirtualUI::AllowExecute Method

Allows the execution of the passed application.

**IDL**

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

<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.

### 10.1.2.6.1.8 IVirtualUI::SetImageQualityByWnd Method

Allows to the the image quality for the specified window.

**IDL**

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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] long Wnd</td>
<td>Window handle.</td>
</tr>
<tr>
<td>[in] BSTR Class</td>
<td>Window classname.</td>
</tr>
<tr>
<td>[in] long Quality</td>
<td>Quality from 0 to 100.</td>
</tr>
</tbody>
</table>

### 10.1.2.6.1.9 IVirtualUI::UploadFile Method

Selects a file from client machine, and it's uploaded to ServerDirectory
### IVirtualUI Methods

#### 10.1.2.6.1.10 IVirtualUI::TakeScreenshot Method

Takes a screenshot of a Window.

**IDL**

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

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] long Wnd</td>
<td>The Window to capture.</td>
</tr>
</tbody>
</table>

#### 10.1.2.6.1.11 IVirtualUI::PreviewPdf Method

Sends the specified PDF file to be shown on the end-user's web browser. It's similar to **PrintPdf**, except that disables the printing options in the browser. Built-in browser printing commands will be available.

**IDL**

```idl
[id(0x0000007A)]
HRESULT _stdcall PreviewPdf([in] BSTR AFileName);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR AFileName</td>
<td>Name of the PDF file.</td>
</tr>
</tbody>
</table>
10.1.2.6.2 IVirtualUI Properties

The properties of the IVirtualUI class are listed here.

### Interface

**IVirtualUI Interface**

### 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 working parameters on the client side.</td>
</tr>
<tr>
<td>Recorder</td>
<td>Controls session recording.</td>
</tr>
<tr>
<td>FileSystemFilter</td>
<td>Controls the filesystem virtualization.</td>
</tr>
<tr>
<td>RegistryFilter</td>
<td>Controls the registry virtualization.</td>
</tr>
<tr>
<td>Options</td>
<td>Get/sets option flags.</td>
</tr>
<tr>
<td>HTMLDoc</td>
<td>Contains methods to modify the behavior on the HTML page.</td>
</tr>
</tbody>
</table>

#### 10.1.2.6.2.1 IVirtualUI::Active Property

Returns the VirtualUI’s state.

**IDL**

```idl
__property VARIANT_BOOL* Active;
```

#### 10.1.2.6.2.2 IVirtualUI::Enabled Property

Enables/disables VirtualUI for the container application.

**IDL**

```idl
__property VARIANT_BOOL Enabled;
```
### Group

**IVirtualUI Properties**

#### 10.1.2.6.2.3 IVirtualUI::DevMode Property

Gets/sets the development mode.

**IDL**

```idl
__property VARIANT_BOOL DevMode;
```

#### 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.

### Group

**IVirtualUI Properties**

#### 10.1.2.6.2.4 IVirtualUI::StdDialogs Property

Enables/disables the use of standard dialogs.

**IDL**

```idl
__property VARIANT_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.

### Group

**IVirtualUI Properties**

#### 10.1.2.6.2.5 IVirtualUI::BrowserInfo Property

Contains information regarding the end-user's environment.

**IDL**

```idl
__property IBrowserInfo** BrowserInfo;
```
10.1.2.6.2.6 IVirtualUI::DevServer Property

Allows for managing the Development Server.

IDL
__property IDevServer** DevServer;

Group
IVirtualUI Properties

10.1.2.6.2.7 IVirtualUI::ClientSettings Property

Controls working parameters on the client side.

IDL
__property IClientSettings** ClientSettings;

Group
IVirtualUI Properties

10.1.2.6.2.8 IVirtualUI::Recorder Property

Controls session recording.

IDL
__property IRecorder** Recorder;

Group
IVirtualUI Properties

10.1.2.6.2.9 IVirtualUI::FileSystemFilter Property

Controls the filesystem virtualization.

IDL
__property IFileSystemFilter** FileSystemFilter;

Group
IVirtualUI Properties

10.1.2.6.2.10 IVirtualUI::RegistryFilter Property

Controls the registry virtualization.

IDL
__property IRegistryFilter** RegistryFilter;
10.1.2.6.2.11 IVirtualUI::Options Property

Get/sets option flags.

**IDL**

```idl
_property unsigned long Options;
```

10.1.2.6.2.12 IVirtualUI::HTMLDoc Property

Contains methods to modify the behavior on the HTML page.

**IDL**

```idl
_property IHTMLDoc** HTMLDoc;
```

10.1.2.7 IBrowserInfo Interface

Interface to 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**

```
IDispatch - Thinfinity::IBrowserInfo
```

IDL

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

**File**

VirtualUIX.ridl

**Library**

Thinfinity Library
10.1.2.7.1 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>GetCookie</td>
<td>Gets the all the browser's cookies.</td>
</tr>
<tr>
<td>SetCookie</td>
<td>Sets a cookie.</td>
</tr>
</tbody>
</table>

### 10.1.2.7.1.1 IBrowserInfo::GetCookie Method

Gets the all the browser's cookies.

**IDL**

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

**Group**

IBrowserInfo Methods

### 10.1.2.7.1.2 IBrowserInfo::SetCookie Method

Sets a cookie.

**IDL**

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

**Group**

IBrowserInfo Methods

### 10.1.2.7.2 IBrowserInfo Properties

The properties of the IBrowserInfo class are listed here.

**Interface**

IBrowserInfo Interface

**Public Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ViewWidth</strong></td>
<td>Returns the width of the VirtualUI Viewer.</td>
</tr>
<tr>
<td><strong>ViewHeight</strong></td>
<td>Returns the height of the VirtualUI Viewer.</td>
</tr>
<tr>
<td><strong>BrowserWidth</strong></td>
<td>Returns the width of the window containing the VirtualUI Viewer.</td>
</tr>
<tr>
<td><strong>BrowserHeight</strong></td>
<td>Returns the height of the window containing the VirtualUI Viewer.</td>
</tr>
<tr>
<td><strong>ScreenWidth</strong></td>
<td>Returns the width of the end-user's monitor screen.</td>
</tr>
<tr>
<td><strong>ScreenHeight</strong></td>
<td>Returns the height of the end-user's monitor screen.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>Returns the logged-on Username.</td>
</tr>
<tr>
<td><strong>IPAddress</strong></td>
<td>Returns the client's IP address.</td>
</tr>
<tr>
<td><strong>UserAgent</strong></td>
<td>Returns the browser's User Agent string.</td>
</tr>
<tr>
<td><strong>ScreenResolution</strong></td>
<td>Returns the application screen resolution defined in the application profile.</td>
</tr>
<tr>
<td><strong>Orientation</strong></td>
<td>Returns the browser's orientation.</td>
</tr>
<tr>
<td><strong>UniqueBrowserId</strong></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><strong>Location</strong></td>
<td>Returns the URL of the current application.</td>
</tr>
<tr>
<td><strong>CustomData</strong></td>
<td>Gets or sets custom user data.</td>
</tr>
<tr>
<td><strong>SelectedRule</strong></td>
<td>Returns the selected Browser Rule.</td>
</tr>
</tbody>
</table>

### 10.1.2.7.2.1 IBrowserInfo::ViewWidth Property

Returns the width of the VirtualUI Viewer.

**IDL**

```idl
__property long ViewWidth;
```

**Group**

IBrowserInfo Properties

### 10.1.2.7.2.2 IBrowserInfo::ViewHeight Property

Returns the height of the VirtualUI Viewer.

**IDL**

```idl
__property long ViewHeight;
```

**Group**

IBrowserInfo Properties

© 2018, Cybele Software, Inc.
10.1.2.7.2.3 IBrowserInfo::BrowserWidth Property

Returns the width of the window containing the VirtualUI Viewer.

IDL
__property long* BrowserWidth;

Group
IBrowserInfo Properties

10.1.2.7.2.4 IBrowserInfo::BrowserHeight Property

Returns the height of the window containing the VirtualUI Viewer.

IDL
__property long* BrowserHeight;

Group
IBrowserInfo Properties

10.1.2.7.2.5 IBrowserInfo::ScreenWidth Property

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

IDL
__property long* ScreenWidth;

Group
IBrowserInfo Properties

10.1.2.7.2.6 IBrowserInfo::ScreenHeight Property

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

IDL
__property long* ScreenHeight;

Group
IBrowserInfo Properties

10.1.2.7.2.7 IBrowserInfo::Username Property

Returns the logged-on Username.

IDL
__property BSTR* Username;
### Group
IBrowserInfo Properties

**10.1.2.7.2.8 IBrowserInfo::IPAddress Property**

Returns the client's IP address.

**IDL**

```idl
__property BSTR* IPAddress;
```

### Group
IBrowserInfo Properties

**10.1.2.7.2.9 IBrowserInfo::UserAgent Property**

Returns the browser's User Agent string

**IDL**

```idl
__property BSTR* UserAgent;
```

### Group
IBrowserInfo Properties

**10.1.2.7.2.10 IBrowserInfo::ScreenResolution Property**

Returns the application screen resolution defined in the application profile.

**IDL**

```idl
__property long* ScreenResolution;
```

### Group
IBrowserInfo Properties

**10.1.2.7.2.11 IBrowserInfo::Orientation Property**

Returns the browser's orientation.

**IDL**

```idl
__property enum Orientation* Orientation;
```
10.1.2.7.2.12 IBrowserInfo::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.

IDL
__property BSTR* UniqueBrowserId;

Group
IBrowserInfo Properties

10.1.2.7.2.13 IBrowserInfo::Location Property

Returns the URL of the current application.

IDL
__property BSTR* Location;

Group
IBrowserInfo Properties

10.1.2.7.2.14 IBrowserInfo::CustomData Property

Gets or sets custom user data.

IDL
__property BSTR CustomData;

Group
IBrowserInfo Properties

10.1.2.7.2.15 IBrowserInfo::SelectedRule Property

Returns the selected Browser Rule.

IDL
__property BSTR* SelectedRule;

Group
IBrowserInfo Properties

10.1.2.8 IDevServer Interface

Allows for managing the Development Server.

Group
Class Hierarchy
IDL
[ uuid(B3EAC0CA-D7AB-4AB1-9E24-84A63C8C3F80), dual, oleautomation ]
interface IDevServer : IDispatch;

File
VirtualUIX.ridl

Library
Thinfinity Library

10.1.2.8.1 IDevServer Properties

The properties of the IDevServer class are listed here.

Interface
IDevServer Interface

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>

10.1.2.8.1.1 IDevServer::Enabled Property

Enables/disables the Development Server.

IDL
__property VARIANT_BOOL Enabled;

Group
IDevServer Properties

10.1.2.8.1.2 IDevServer::Port Property

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

IDL
__property long Port;

Group
IDevServer Properties

10.1.2.8.1.3 IDevServer::StartBrowser Property

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

**IDL**

```idl
__property VARIANT_BOOL StartBrowser;
```

Group

IDevServer Properties

10.1.2.9 IClientSettings Interface

Controls working parameters on the client side.

**Class Hierarchy**

```
IDispatch  Thinity:IClientSettings
```

**File**

VirtualUIX.ridl

**Library**

Thinity Library

10.1.2.9.1 IClientSettings Properties

The properties of the IClientSettings class are listed here.

**Interface**

IClientSettings Interface

**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>
</tbody>
</table>
10.1.2.9.1.1 IClientSettings::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).

IDL
__property enum MouseMoveGestureStyle MouseMoveGestureStyle;

Group
IClientSettings Properties

10.1.2.9.1.2 IClientSettings::MouseMoveGestureAction Property

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

IDL
__property enum MouseMoveGestureAction MouseMoveGestureAction;

Group
IClientSettings Properties

10.1.2.9.1.3 IClientSettings::CursorVisible Property

Hides/shows the mouse pointer.

IDL
__property VARIANT_BOOL CursorVisible;

Group
IClientSettings Properties

10.1.2.1 IJSValue Interface

Base interface for IJSNamedValue, IJSPROPERTY and IJSArgument.

Class Hierarchy

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

The properties of the IJSValue class are listed here.

### Interface

**IJSValue Interface**

### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Type</strong></td>
<td>Sets/gets a data type.</td>
</tr>
<tr>
<td><strong>RawValue</strong></td>
<td>Sets/gets a raw value.</td>
</tr>
<tr>
<td><strong>AsString</strong></td>
<td>Sets/gets a string value.</td>
</tr>
<tr>
<td><strong>AsInt</strong></td>
<td>Sets/gets an integer value.</td>
</tr>
<tr>
<td><strong>AsBool</strong></td>
<td>Sets/gets a boolean value.</td>
</tr>
<tr>
<td><strong>AsFloat</strong></td>
<td>Sets/gets a float value.</td>
</tr>
<tr>
<td><strong>AsJSON</strong></td>
<td>Sets/gets a JSON-formatted string value.</td>
</tr>
</tbody>
</table>

#### 10.1.2.10.1.1 IJSValue::DataType Property

Sets/gets a data type.

```idl
__property enum IJSDataType DataType;
```

#### 10.1.2.10.1.2 IJSValue::RawValue Property

Sets/gets a raw value.

```idl
__property VARIANT RawValue;
```
10.1.2.10.1.3 IJSValue::AsString Property

Sets/gets a string value.

IDL
__property BSTR AsString;

10.1.2.10.1.4 IJSValue::AsInt Property

Sets/gets an integer value.

IDL
__property long AsInt;

10.1.2.10.1.5 IJSValue::AsBool Property

Sets/gets a boolean value.

IDL
__property VARIANT_BOOL AsBoolean;

10.1.2.10.1.6 IJSValue::AsFloat Property

Sets/gets a float value.

IDL
__property float AsFloat;
10.1.2.10.1.7 IJSValue::AsJSON Property

Sets/gets a JSON-formatted string value.

```idl
__property BSTR AsJSON;
```

Group

IJSValue Properties

10.1.2.1 IJSNamedValue Interface

Base interface for IJSProperty and IJSArgument.

Class Hierarchy

```
ID Dispatch ─ Thinfinity:IJSValue ─ Thinfinity:IJSNamedValue
```

IDL

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

File

VirtualUIX.ridl

Library

Thinfinity Library

10.1.2.11.1 IJSNamedValue Properties

The properties of the IJSNamedValue class are listed here.

Interface

IJSNamedValue Interface

Public Properties

<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>

10.1.2.11.1.1 IJSNamedValue::Name Property

Sets/gets the name.

IDL
10.1.2.1 IJSObject Interface

Represents a custom remotable object.

### Class Hierarchy

[uuid(59342310-79A7-4B14-8B63-6DF05609AE30), dual, oleautomation]

```plaintext
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 viceversa, 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

Thinfinity Library

10.1.2.12.1 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>FireEvent</td>
<td>Fires the event specified in Name on the client-size javascript</td>
</tr>
</tbody>
</table>
10.1.2.12.1.1 IJSObject::FireEvent Method

Fires the event specified in Name on the client-size javascript API.

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

**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>[in] IJSArguments* Arguments</td>
<td>List of arguments</td>
</tr>
</tbody>
</table>

**Group**
IJSObject Methods

10.1.2.12.1.2 IJSObject::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.

**IDL**
[\id(0x000000CF)]
HRESULT _stdcall ApplyChanges();

**Group**
IJSObject Methods

10.1.2.12.1.3 IJSObject::ApplyModel Method

Propagates the whole JObject definition to the javascript client.

**IDL**
[\id(0x000000D0)]
HRESULT _stdcall ApplyModel();

**Group**
IJSObject Methods
10.1.2.12.2 IJSObject Properties

The properties of the IJSObject class are listed here.

**Interface**

IJSObject Interface

---

**Public Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Id</strong></td>
<td>Identifier of the object. It must be unique among siblings objects.</td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td>List containing all properties of this object.</td>
</tr>
<tr>
<td><strong>Methods</strong></td>
<td>List containing all methods of this object.</td>
</tr>
<tr>
<td><strong>Events</strong></td>
<td>List containing all events of this object.</td>
</tr>
<tr>
<td><strong>Objects</strong></td>
<td>List containing all events of this object.</td>
</tr>
</tbody>
</table>

10.1.2.12.2.1 IJSObject::Id Property

Identifier of the object. It must be unique among siblings objects.

**IDL**

```idl
__property BSTR Id;
```

**Group**

IJSObject Properties

10.1.2.12.2.2 IJSObject::Properties Property

List containing all properties of this object.

**IDL**

```idl
__property IJSProperties** Properties;
```

**Group**

IJSObject Properties

10.1.2.12.2.3 IJSObject::Methods Property

List containing all methods of this object.

**IDL**

```idl
__property IJSMethods** Methods;
```
10.1.2.12.2.4 IJSObject::Events Property

List containing all events of this object.

IDL
__property IJSEvents** Events;

10.1.2.12.2.5 IJSObject::Objects Property

List containing all events of this object.

IDL
__property IJSObjects** Objects;

10.1.2.1 IJSObjects Interface

Collection of IJSObjects

Class Hierarchy

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

File

VirtualUIX.ridl

Library

Thinfinity Library

10.1.2.13.1 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>Item</td>
<td>Returns an item.</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the item.</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>

#### 10.1.2.13.1.1 IJSObjects::Item Method

Returns an item.

**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**

#### 10.1.2.13.1.2 IJSObjects::Clear Method

Clears the collection.

**IDL**

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

#### Group

**IJSObjects Methods**

#### 10.1.2.13.1.3 IJSObjects::Add Method

Adds a new object to the collection.
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**

IJSObjects Methods

10.1.2.13.1.4 IJSObjects::Remove Method

Removes an item from the collection.

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

- **Parameters**

<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

10.1.2.13.2 IJSObjects Properties

The properties of the IJSObjects class are listed here.

- **Interface**

IJSObjects Interface

- **Public Properties**

<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>
</tbody>
</table>

10.1.2.13.2.1 IJSObjects::Count Property

Returns the number of items in the collection.
__property long* Count;

Group
IJSObjects Properties

10.1.2.1 IJSProperty Interface

Represents a property in a custom remotable object.

Class Hierarchy

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

File
VirtualUIX.ridl

Library
Thinfinity Library

10.1.2.14.1 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>
<tbody>
<tr>
<td>OnGet</td>
<td>Allows you to pass an interface that will receive control at property initialization and when you make a call to IJSObject.ApplyChanges.</td>
</tr>
<tr>
<td>OnSet</td>
<td>Allows you to pass an interface that will receive control when a property changes on the client side.</td>
</tr>
</tbody>
</table>

10.1.2.14.1.1 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
[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**

10.1.2.14.1.2 IJSProperty::OnSet Method

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

**IDL**

[0x00000193]

HRESULT _stdcall OnSet([in] IJSBinding* Binding, [out, retval] IJSProperty** OutRetVal);

<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**

10.1.2.14.2 IJSProperty Properties

The properties of the IJSProperty class are listed here.

### Interface

**IJSProperty Interface**

### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadOnly</td>
<td>Sets/get the readonly attribute. If Readonly is true, the property value cannot be altered on the client side.</td>
</tr>
</tbody>
</table>

10.1.2.14.2.1 IJSProperty::ReadOnly Property

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

Group
IJSProperty Properties

10.1.2.1 IJSProperties Interface
Collection of properties.

Class Hierarchy
IDL
[ uuid(FCBB688F-8FB2-42C1-86FC-0A9F3B2A500C), dual, oleautomation ]
interface IJSProperties : IDispatch;

File
VirtualUIX.ridl

Library
Thinfinity Library

10.1.2.15 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>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>

10.1.2.15.1 IJSProperties::Item Method
Returns an item.

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

<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**

**IJSProperties Methods**

10.1.2.15.1.2 IJSProperties::Clear Method

Clears the collection.

**IDL**

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

**Group**

**IJSProperties Methods**

10.1.2.15.1.3 IJSProperties::Add Method

Adds a new property to the collection.

**IDL**

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

<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** OutRetVal</td>
<td>Returns the newly created property.</td>
</tr>
</tbody>
</table>

**Group**

**IJSProperties Methods**

10.1.2.15.1.4 IJSProperties::Remove Method

Removes an item from the collection.

**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>Item to remove</td>
</tr>
</tbody>
</table>

### Group

IJSProperties Methods

10.1.2.15.2 IJSProperties Properties

The properties of the IJSProperties class are listed here.

### Interface

IJSProperties Interface

### Public Properties

<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>
</tbody>
</table>

10.1.2.15.2.1 IJSProperties::Count Property

Returns the number of items in the collection.

**IDL**

```idl
__property long* Count;
```

### Group

IJSProperties Properties

10.1.2.1 IJSArgument Interface

6

Represents a parameter in events and methods.

### Class Hierarchy

```
IDispatch ➔ Thinfinity:ISValue ➔ Thinfinity:ISNamedValue ➔ Thinfinity:ISArgument
```

**IDL**

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

### File

VirtualUIX.ridl
10.1.2.1 IJSArguments Interface

Collection of IJSArgument.

## Class Hierarchy

```
interface IJSArguments : IDispatch;
```

## File

VirtualUIX.ridl

### Library

Thinfinity Library

10.1.2.17.1 IJSArguments Methods

The methods of the IJSArguments class are listed here.

## Interface

IJSArguments Interface

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>

### 10.1.2.17.1.1 IJSArguments::Item Method

Returns an item of the collection.

```
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

### 10.1.2.17.1.2 IJSArguments::Clear Method

Clears the collection.

**IDL**

[id(0x000000CC)]

HRESULT _stdcall Clear();

## Group

IJSArguments Methods

### 10.1.2.17.1.3 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 IJSArgument</td>
</tr>
</tbody>
</table>

## Group

IJSArguments Methods

### 10.1.2.17.1.4 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>Item IJSArgument*</td>
<td>IJSMethod to remove</td>
</tr>
</tbody>
</table>

## Group

IJSArguments Methods

### 10.1.2.17.2 IJSArguments Properties

The properties of the IJSArguments class are listed here.

## Interface

IJSArguments Interface

## Public Properties

<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>
</tbody>
</table>

### 10.1.2.17.2.1 IJSArguments::Count Property

Returns the number of items in the collection.

**IDL**

```idl
__property long* Count;
```

## Group

IJSArguments Properties

### 10.1.2.1 IJSMethod Interface

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

## Class Hierarchy

```
IDispatch  Thinfinity.IJSMethod
```

**IDL**

```idl
[ uuid(C45D6A8F-AD4A-47BB-AC3A-C125D6D5D27E), dual, oleautomation ]
interface IJSMethod : IDispatch;
```

## File

VirtualUIX.ridl

© 2018, Cybele Software, Inc.
10.1.2.18.1 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>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>

10.1.2.18.1.1 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>

10.1.2.18.1.2 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**
### 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**

### 10.1.2.18.2 IJSMETHOD Properties

The properties of the IJSMETHOD class are listed here.

#### Interface

**IJSMETHOD Interface**

#### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ] Name</td>
<td>This is Name, a member of class IJSMETHOD.</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>
</tbody>
</table>

#### 10.1.2.18.2.1 IJSMETHOD::Name Property

**IDL**

```idl
__property BSTR Name;
```

#### Description

This is Name, a member of class IJSMETHOD.

#### Group

**IJSMETHOD Properties**

#### 10.1.2.18.2.2 IJSMETHOD::Arguments Property

Collection of arguments

**IDL**

```idl
__property IJSarguments** Arguments;
```
10.1.2.18.2.3 IJSMethod::ReturnValue Property

Data to return to the remote caller.

IDL
__property IJSValue** ReturnValue;

Group
IJSMethod Properties

10.1.2.1 IJSMethods Interface

Collection of methods.

Class Hierarchy

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

File
VirtualUIX.ridl

Library
Thinfinity Library

10.1.2.19.1 IJSMethods Methods

The methods of the IJSMethods class are listed here.

Interface
IJSMethods Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
10.1.2.19.1.1 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>

**Group**

IJSMethods Methods

10.1.2.19.1.2 IJSMethods::Clear Method

Clears the collection.

**IDL**

[id(0x000000cc)]

HRESULT _stdcall Clear();

**Group**

IJSMethods Methods

10.1.2.19.1.3 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>

**Group**

IJSMethods Methods
10.1.2.19.1.4 IJSMethods::Remove Method

Removes an item from the collection.

**IDL**

```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>

**Group**

IJSMethods Methods

10.1.2.19.2 IJSMethods Properties

The properties of the IJSMethods class are listed here.

**Interface**

IJSMethods Interface

**Public Properties**

<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>
</tbody>
</table>

10.1.2.19.2.1 IJSMethods::Count Property

Returns the number of items in the collection.

**IDL**

```idl
__property long* Count;
```

**Group**

IJSMethods Properties

10.1.2.2 IJSEvent Interface

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

**Class Hierarchy**

[Diagram: IDispatch → Thinfinity::IJSEvent]
IDL
[ uuid(8B66EACD-9619-43CF-9196-DCDA17F5500E), dual, oleautomation ]
interface IJSEvent : IDispatch;

File
VirtualUIX.ridl

Library
Thinfinity Library

10.1.2.20.1 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>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>

10.1.2.20.1.1 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>
</tbody>
</table>
### IJSEvent Methods

#### 10.1.2.20.1.2 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>

#### 10.1.2.20.1.3 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);
```

**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.</td>
</tr>
<tr>
<td>[out, retval] IJSEvent** OutRetVal</td>
<td>Returns the parent IJSEvent.</td>
</tr>
</tbody>
</table>

#### 10.1.2.20.1.4 IJSEvent::ArgumentAsInt Method

Sets the value of an argument with a integer data type.
### 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] 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**

#### 10.1.20.1.5 IJSEvent::ArgumentAsBool Method

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

### 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] 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>

### Group

**IJSEvent Methods**

#### 10.1.20.1.6 IJSEvent::ArgumentAsFloat Method

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

### 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>
</tbody>
</table>
Groups

IJSEvent Methods

10.1.2.20.1.7 IJSEvent::ArgumentAsJSON Method

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

**IDL**

```
[id(0x000000D1)]
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>

Groups

IJSEvent Methods

10.1.2.20.1.8 IJSEvent::Fire Method

Fires the event.

**IDL**

```
[id(0x000000D2)]
HRESULT _stdcall Fire();
```

Groups

IJSEvent Methods

10.1.2.20.2 IJSEvent Properties

The properties of the IJSEvent class are listed here.

Interface

IJSEvent Interface

Public Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>This is Name, a member of class IJSEvent.</td>
</tr>
<tr>
<td>Arguments</td>
<td>This is Arguments, a member of class IJSEvent.</td>
</tr>
</tbody>
</table>

### 10.1.2.20.2.1 IJSEvent::Name Property

**IDL**

```
__property BSTR Name;
```

**Description**

This is Name, a member of class IJSEvent.

**Group**

IJSEvent Properties

### 10.1.2.20.2.2 IJSEvent::Arguments Property

**IDL**

```
__property IJSArguments** Arguments;
```

**Description**

This is Arguments, a member of class IJSEvent.

**Group**

IJSEvent Properties

### 10.1.2.2 IJSEvents Interface

Collection of events.

**Class Hierarchy**

```
IDispatch  ─ Thinfinity::IJSEvents
```

**IDL**

```
[ uuid(6AE952B3-B6DA-4C81-80FF-D0A162E11D02), dual, oleautomation ]
interface IJSEvents : IDispatch;
```

**File**

VirtualUIX.ridl

**Library**

Thinfinity Library
10.1.2.21.1 IJSEvents Methods

The methods of the IJSEvents class are listed here.

### Interface

IJSEvents Interface

### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 an IJSEvent to the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an item from the collection.</td>
</tr>
</tbody>
</table>

10.1.2.21.1.1 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

10.1.2.21.1.2 IJSEvents::Clear Method

Clears the collection.

**IDL**

[id(0x000000CC)]

HRESULT _stdcall Clear();

### Group

IJSEvents Methods
10.1.2.21.1.3  IJSEvents::Add Method

Adds an IJSEvent to the collection.

**IDL**

```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

10.1.2.21.1.4  IJSEvents::Remove Method

Removes an item from the collection.

**IDL**

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

**Parameters**

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

**Group**

IJSEvents Methods

10.1.2.21.2  IJSEvents Properties

The properties of the IJSEvents class are listed here.

**Interface**

IJSEvents Interface

**Public Properties**

<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>
</tbody>
</table>
10.1.2.21.2.1 IJSEvents::Count Property

Returns the number of items in the collection.

IDL

```idl
__property long* Count;
```

Group

IJSEvents Properties

10.1.2.2 IJSBinding Interface

Used as a callback interface for binding external data source with the IJSPROPERTY data.

Class Hierarchy

IDL

```idl
[ uuid(ACFC2953-37F1-479E-B405-D0BB75E156E6), dual, oleautomation ]
interface IJSBinding : IDispatch;
```

File

VirtualUIX.ridl

Library

Thinfinity Library

10.1.2.22.1 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>

10.1.2.22.1.1 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>Parent IJSObject</td>
<td>Parent IJSObject</td>
</tr>
<tr>
<td>Calling IJSPROPERTY</td>
<td>Calling IJSPROPERTY</td>
</tr>
</tbody>
</table>

### Group

IJSCallback Methods

10.1.2.2 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

Thinfinity Library

10.1.2.23.1 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>

10.1.2.23.1.1 IJSCallback::Callback Method

Callback method.
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

#### 10.1.2.2 IRecorder Interface

Controls session recording.

### Class Hierarchy

[ IDispatch ➔ Thinfinity::IRecorder ]

IDL
[ uuid(D89DA2B6-B7BF-4065-80F5-6D78B331C7DD), dual, oleautomation ]
interface IRecorder : IDispatch;

### File

VirtualUIX.ridl

### Library

Thinfinity Library

#### 10.1.2.24.1 IRecorder Methods

The methods of the IRecorder class are listed here.

### Interface

IRecorder Interface

### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rec</td>
<td>Starts the session recording.</td>
</tr>
</tbody>
</table>
10.1.24.1.1 IRecorder::Rec Method

Starts the session recording.

**IDL**

```plaintext
[id(0x000000CB)]
HRESULT _stdcall Rec([in] BSTR Track);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] BSTR Track</td>
<td>Track name. In the same session file, there may be one or more tracks.</td>
</tr>
</tbody>
</table>

**Group**

IRecorder Methods

10.1.24.1.2 IRecorder::Play Method

Starts the playback of the loaded session file.

**IDL**

```plaintext
[id(0x000000CC)]
HRESULT _stdcall Play([in] long From, [in] long To);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] long From</td>
<td>Entry of playback start.</td>
</tr>
<tr>
<td>[in] long To</td>
<td>Last entry to play.</td>
</tr>
</tbody>
</table>

**Description**

**Remarks**

To play a recorded session:

- Set the Filename property with the path and name of the session file to be played (the file with "idx" extension).

- Call Play with the range of entries to play.

To play an entire session pass 0 and the Count property minus 1, or -1 in both parameters.

To play only a specific track, pass the Position of track to reproduce as From, and the Position of the next track minus 1 as To. For the last track, the To parameter must be the Count property of recorder minus 1.
10.1.24.1.3 IRecorder::Stop Method

Stops the session recording started by Rec(), or playback started by Play().

IDL

```
[id(0x000000CD)]
HRESULT _stdcall Stop();
```

Description

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>Gets or sets the path and filename to be used for session recording or playback.</td>
</tr>
<tr>
<td>Position</td>
<td>During playback, returns the session entry that is playing</td>
</tr>
<tr>
<td>Count</td>
<td>Returns the total number of entries in the session file loaded for playback.</td>
</tr>
<tr>
<td>Options</td>
<td>This is Options, a member of class IRecorder.</td>
</tr>
<tr>
<td>Tracks</td>
<td>Collection of tracks (IRecTrack) contained in the loaded session file.</td>
</tr>
</tbody>
</table>

10.1.24.2.1 IRecorder::Filename Property

Gets or sets the path and filename to be used for session recording or playback.
**Remarks**

For recording: the path and filename where the session will be stored. No extension is needed: the Recorder will create two files with extensions ".dat" and ".idx".

For playback: a existing file to be loaded (the file with "idx" extension).

**Group**

IRecorder Properties

**10.1.24.2.2 IRecorder::Position Property**

During playback, returns the session entry that is playing

IDL

```idl
__property long* Position;
```

**Description**

**Group**

IRecorder Properties

**10.1.24.2.3 IRecorder::Count Property**

Returns the total number of entries in the session file loaded for playback.

IDL

```idl
__property long* Count;
```

**Description**

**Group**

IRecorder Properties

**10.1.24.2.4 IRecorder::State Property**

Returns the Recorder state:

Inactive (0): The Recorder is idle.
Recording (1): Recording of a session in progress.
__property enum RecorderState* State;

Description

Group

IRecorder Properties

10.1.2.24.2.5 IRecorder::Options Property

IDL
__property unsigned long Options;

Description

This is Options, a member of class IRecorder.

Group

IRecorder Properties

10.1.2.24.2.6 IRecorder::Tracks Property

Collection of tracks (IRecTrack) contained in the loaded session file.

IDL
__property IRecTracks** Tracks;

Description

Group

IRecorder Properties

10.1.2 IRecTrack Interface

Track of recorded session.

Class Hierarchy

[ uuid(D4744AE1-70CB-43DD-BEA5-A5310B2E24C6), dual, oleautomation ]
interface IRecTrack : IDispatch;

File

VirtualUIX.ridl

© 2018, Cybele Software, Inc.
10.1.2.25.1 IRecTrack Properties

The properties of the IRecTrack class are listed here.

### Interface

**IRecTrack Interface**

### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Returns the Track name, specified in the Recorder.Rec() method.</td>
</tr>
<tr>
<td>Position</td>
<td>Returns the entry index in the entire session file where this track starts.</td>
</tr>
</tbody>
</table>

#### 10.1.2.25.1.1 IRecTrack::Name Property

Returns the Track name, specified in the Recorder.Rec() method.

**IDL**

```idl
__property BSTR* Name;
```

#### 10.1.2.25.1.2 IRecTrack::Position Property

Returns the entry index in the entire session file where this track starts.

**IDL**

```idl
__property long* Position;
```
10.1.2.2 IRecTracks Interface

6

Collection of IRecTrack. Tracks contained in a recorded session.

Class Hierarchy

IDL

[ uuid(AB45B615-9309-471E-A455-3FE93F88E674), dual, oleautomation ]

interface IRecTracks : IDispatch;

File

VirtualUIX.ridl

Description

Library

Thinfinity Library

10.1.2.26.1 IRecTracks Methods

The methods of the IRecTracks class are listed here.

Interface

IRecTracks Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Returns a track of the collection.</td>
</tr>
</tbody>
</table>

10.1.2.26.1.1 IRecTracks::Item Method

Returns a track of the collection.

IDL

[propget, id(0x000000C9)]

HRESULT _stdcall Item([in] long Index, [out, retval] IRecTrack** Value);

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[in] long Index</td>
<td>Index of the track to get, between 0 and Count.</td>
</tr>
<tr>
<td>OutRetVal</td>
<td>Track in specified Index.</td>
</tr>
</tbody>
</table>
Description

Group

IRecTracks Methods

10.1.2.26.2 IRecTracks Properties

The properties of the IRecTracks class are listed here.

Interface

IRecTracks Interface

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Returns the count of tracks in the collection.</td>
</tr>
</tbody>
</table>

10.1.2.26.2.1 IRecTracks::Count Property

Returns the count of tracks in the collection.

IDL

__property long* Count;

Description

Group

IRecTracks Properties

10.1.2.2 IFileSystemFilter Interface

Controls the filesystem virtualization.

Class Hierarchy

IDL

[ uuid(3FE99D2F-0CFC-43D1-B762-0C7C15EB872E), dual, oleautomation ]

interface IFileSystemFilter : IDispatch;

File

VirtualUIX.ridl
10.1.2.27.1  IFileSystemFilter Properties

The properties of the IFileSystemFilter class are listed here.

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Gets or sets the user to which this filter will be applied.</td>
</tr>
<tr>
<td>CfgFile</td>
<td>Gets or sets the configuration file.</td>
</tr>
<tr>
<td>Active</td>
<td>Gets or sets the active state.</td>
</tr>
</tbody>
</table>

10.1.2.27.1.1  IFileSystemFilter::User Property

Gets or sets the user to which this filter will be applied.

IDL

```idl
__property BSTR User;
```

10.1.2.27.1.2  IFileSystemFilter::CfgFile Property

Gets or sets the configuration file.

IDL

```idl
__property BSTR CfgFile;
```

10.1.2.27.1.3  IFileSystemFilter::Active Property

Gets or sets the active state.

IDL

```idl
__property VARIANT_BOOL Active;
```
10.1.2.2 IRegistryFilter Interface

Controls the registry virtualization.

### Class Hierarchy

```idl
[ uuid(4834F840-915B-488B-ADEA-98890A04CEE6), dual, oleautomation ]
interface IRegistryFilter : IDispatch;
```

### File

VirtualUIX.ridl

### Library

Thinfinity Library

10.1.2.28.1 IRegistryFilter Properties

The properties of the IRegistryFilter class are listed here.

#### Interface

IRegistryFilter Interface

#### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Gets or sets the user to which this filter will be applied.</td>
</tr>
<tr>
<td>CfgFile</td>
<td>Gets or sets the configuration file.</td>
</tr>
<tr>
<td>Active</td>
<td>Gets or sets the active state.</td>
</tr>
</tbody>
</table>

10.1.2.28.1.1 IRegistryFilter::User Property

Gets or sets the user to which this filter will be applied.

```idl
__property BSTR User;
```
IRegistryFilter Properties

10.1.2.28.1.2 IRegistryFilter::CfgFile Property

Gets or sets the configuration file.

IDL
__property BSTR CfgFile;

Group
IRegistryFilter Properties

10.1.2.28.1.3 IRegistryFilter::Active Property

Gets or sets the active state.

IDL
__property VARIANT_BOOL Active;

Group
IRegistryFilter Properties

10.1.2.2 IHTMLDoc Interface

Contains methods to modify the behavior on the HTML page.

Class Hierarchy

IDL
[ uuid(A5A7F58C-D83C-4C89-872E-0C51A9B5D3B0), dual, oleautomation ]
interface IHTMLDoc : IDispatch;

File
VirtualUIX.ridl

Library
Thinfinity Library

10.1.2.29.1 IHTMLDoc Methods

The methods of the IHTMLDoc class are listed here.
### IHTMLDoc Interface

#### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateSessionURL</td>
<td>Creates an url pointing to a local filename. This url is valid during the session lifetime and its private to this session.</td>
</tr>
<tr>
<td>CreateComponent</td>
<td>Inserts an HTML. Used to insert regular HTML elements or WebComponents with custom elements.</td>
</tr>
<tr>
<td>GetSafeURL</td>
<td>Gets a safe, temporary and unique URL to access any file in the disk.</td>
</tr>
<tr>
<td>LoadScript</td>
<td>Loads a script from url.</td>
</tr>
<tr>
<td>ImportHTML</td>
<td>Imports an HTML from URL. If Filename is specified, creates a session URL first and then imports the html file from that Filename.</td>
</tr>
</tbody>
</table>

10.1.29.1.1 IHTMLDoc::CreateSessionURL Method

Creates an url pointing to a local filename. This url is valid during the session lifetime and its private to this session.

**IDL**

```c
[id(0x000000CC)]
HRESULT __stdcall CreateSessionURL([in] BSTR Url, [in] BSTR Filename);
```

**Group**

IHTMLDoc Methods

10.1.29.1.2 IHTMLDoc::CreateComponent Method

Inserts an HTML. Used to insert regular HTML elements or WebComponents with custom elements.

**IDL**

```c
[id(0x000000CD)]
HRESULT __stdcall CreateComponent([in] BSTR Id, [in] BSTR Html, [in] __int64 ReplaceWnd);
```

**Group**

IHTMLDoc Methods

10.1.29.1.3 IHTMLDoc::GetSafeURL Method

Gets a safe, temporary and unique URL to access any file in the disk.

**IDL**

```c
[id(0x000000CE)]
HRESULT __stdcall GetSafeURL([in] BSTR Filename, [in] long Minutes, [out, retval] BSTR* OutRetVal);
```
10.1.2.29.1.4 IHTMLDoc::LoadScript Method

Loads a script from url.

**IDL**

```idl
HRESULT __stdcall LoadScript([in] BSTR Url, [in, optional] BSTR Filename);
```

10.1.2.29.1.5 IHTMLDoc::ImportHTML Method

Imports an HTML from URL. If Filename is specified, creates a session URL first and then imports the html file from that Filename.

**IDL**

```idl
HRESULT __stdcall ImportHTML([in] BSTR Url, [in, optional] BSTR Filename);
```

10.1.2.3 IEvents Interface

General VirtualUI events.

**Class Hierarchy**

```idl
[ uuid(1C5700BC-2317-4062-B614-0A4E286CFE68) ]
<dispinterface> IEvents;
```

**File**

VirtualUIX.ridl

**Library**

Thinfinity Library
10.1.2.30.1 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>This is OnDownloadEnd, a member of class IEvents.</td>
</tr>
<tr>
<td>OnRecorderChanged</td>
<td>Fires when there is a change in the recording or playback status.</td>
</tr>
</tbody>
</table>

#### 10.1.2.30.1.1 IEvents::OnGetUploadDir Method

Fires during an upload request, allowing you to change the save folder.

**IDL**

```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>

#### 10.1.2.30.1.2 IEvents::OnBrowserResize Method

Fires when the VirtualUI Viewer's container window resizes. Normally, when the browser resizes.

**IDL**

```idl
[id(0x00000066)]
void OnBrowserResize([in, out] long* Width, [in, out] long* Height, [in, out] VARIANT_BOOL* Resize);
```
### 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 resizing all maximized windows.

### Group

**IEvents Methods**

#### 10.1.2.30.1.3 IEvents::OnClose Method

Fires when the browser window is about to close.

**IDL**

```idl
[id(0x00000067)]
void OnClose();
```

### Group

**IEvents Methods**

#### 10.1.2.30.1.4 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**
10.1.2.30.1.5 IEvents::OnDownloadEnd Method

**IDL**

```
[id(0x00000069)]
void OnDownloadEnd([in] BSTR Filename);
```

**Description**

This is OnDownloadEnd, a member of class IEvents.

**Group**

IEvents Methods

10.1.2.30.1.6 IEvents::OnRecorderChanged Method

Fires when there is a change in the recording or playback status.

**IDL**

```
[id(0x0000006A)]
HRESULT OnRecorderChanged();
```

**Group**

IEvents Methods

10.1.2.3 IJSObjectEvents Interface

Interface to IJSObject events.

**Class Hierarchy**

```
[ uuid(A3D640E8-CD18-4196-A1A2-C87C82B0F88B) ]
dispinterface IJSObjectEvents;
```

**File**

VirtualUIX.ridl

**Library**

Thinfinity Library

10.1.2.31.1 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>
<tbody>
<tr>
<td><strong>OnExecuteMethod</strong></td>
<td>Fires when a method is called in the client side.</td>
</tr>
<tr>
<td><strong>OnPropertyChange</strong></td>
<td>Fires when a property has changed on the client side.</td>
</tr>
</tbody>
</table>

### 10.1.2.31.1.1  IJSObjectEvents::OnExecuteMethod Method

Fires when a method is called in the client side.

**IDL**

```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**

### 10.1.2.31.1.2  IJSObjectEvents::OnPropertyChange Method

Fires when a property has changed on the client side.

**IDL**

```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

© 2018, Cybele Software, Inc.
10.2 .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>

#### 10.2.1 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 browser window on the end-user's application page.</td>
</tr>
<tr>
<td>HTMLDoc</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>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>VirtualUI</td>
<td>Main class. Has methods, properties and events to allow the activation and control the behavior of VirtualUI.</td>
</tr>
<tr>
<td>JSObject</td>
<td>Represents a custom remotable object.</td>
</tr>
<tr>
<td>JSBinding</td>
<td>This is class Cybele.Thinfinity.JSBinding.</td>
</tr>
<tr>
<td>JSCallback</td>
<td>This is class Cybele.Thinfinity.JSCallback.</td>
</tr>
</tbody>
</table>

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecorderState</td>
<td>This is record Cybele.Thinfinity.RecorderState.</td>
</tr>
</tbody>
</table>

### Group

.NET Classes
10.2.1.1 Cybele.Thinfinity.RecorderState Enumeration

```csharp
public enum RecorderState {
    Inactive = 0,
    Recording = 1,
    Playing = 2
}
```

File
Thinfinity.VirtualUI.cs

Description
This is record Cybele.Thinfinity.RecorderState.

Namespace
Cybele.Thinfinity Namespace

10.2.1.2 ClientSettings Class

Allows to set some client settings.

Class Hierarchy
```
IClientSettings

Disposable

Cybele.Thinfinity.ClientSettings
```

```csharp
public class ClientSettings : IClientSettings, IDisposable;
```

File
Thinfinity.VirtualUI.cs

Namespace
Cybele.Thinfinity Namespace

10.2.1.2.1 ClientSettings.ClientSettings Constructor

```csharp
public ClientSettings(IVirtualUI virtualUI);
```

Description
This is ClientSettings, a member of class ClientSettings.
10.2.1.2.2 ClientSettings Methods

The methods of the ClientSettings class are listed here.

### 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 ClientSettings.</td>
</tr>
</tbody>
</table>

10.2.1.2.2.1 ClientSettings.Dispose Method

**C#**

```csharp
public void Dispose();
```

**Description**

This is Dispose, a member of class ClientSettings.

10.2.1.2.3 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</td>
</tr>
<tr>
<td></td>
<td>acts on the exact spot of the finger touch (absolute) or its position is</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>interpreted as a mouse move or as a mouse wheel.</td>
</tr>
</tbody>
</table>
10.2.1.2.3.1 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;
```

**Group**

ClientSettings Properties

10.2.1.2.3.2 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;
```

**Group**

ClientSettings Properties

10.2.1.2.3.3 ClientSettings.CursorVisible Property

Hides/shows the mouse pointer.

**C#**

```csharp
public bool CursorVisible;
```

**Group**

ClientSettings Properties

10.2.1.3 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**

```csharp
public class BrowserInfo : IBrowserInfo, IDisposable;
```

© 2018, Cybele Software, Inc.
10.2.1.3.1 BrowserInfo.BrowserInfo Constructor

C#
public BrowserInfo(IVirtualUI virtualUI);

Description
This is BrowserInfo, a member of class BrowserInfo.

10.2.1.3.2 BrowserInfo Methods

The methods of the BrowserInfo class are listed here.

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>

10.2.1.3.2.1 BrowserInfo.Dispose Method

C#
public void Dispose();

Description
This is Dispose, a member of class BrowserInfo.
BrowserInfo Methods

10.2.1.3.2.2 BrowserInfo.GetCookie Method

Returns a browser's cookie value.

**C#**

```csharp
public string GetCookie(string Name);
```

Group

BrowserInfo Methods

10.2.1.3.2.3 BrowserInfo.SetCookie Method

Sets a cookie in the browser.

**C#**

```csharp
public void SetCookie(string Name, string Value, string Expires);
```

Group

BrowserInfo Methods

10.2.1.3.3 BrowserInfo Properties

The properties of the BrowserInfo class are listed here.

Class

BrowserInfo 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>
</tbody>
</table>
### BrowserInfo.ViewWidth Property

Returns the width of the VirtualUI Viewer.

```csharp
public int ViewWidth;
```

### BrowserInfo.ViewHeight Property

Returns the height of the VirtualUI Viewer.

```csharp
public int ViewHeight;
```

### BrowserInfo.BrowserWidth Property

```
c#
public int BrowserWidth;

**Description**

Returns the width of the HTML element containing the VirtualUI Viewer.

**Group**

BrowserInfo Properties
10.2.1.3.3.4 BrowserInfo.BrowserHeight Property

Returns the height of the HTML element containing the VirtualUI Viewer.

```csharp
public int BrowserHeight;
```

10.2.1.3.3.5 BrowserInfo.ScreenWidth Property

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

```csharp
public int ScreenWidth;
```

10.2.1.3.3.6 BrowserInfo.ScreenHeight Property

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

```csharp
public int ScreenHeight;
```

10.2.1.3.3.7 BrowserInfo.Username Property

Returns the logged-on Username.

```csharp
public string Username;
```

10.2.1.3.3.8 BrowserInfo.IPAddress Property

Returns the client's IP address.

```csharp
public string IPAddress;
```
10.2.1.3.3.9 BrowserInfo.UserAgent Property

Returns the browser's User Agent string.

C#
public string UserAgent;

10.2.1.3.3.10 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;

10.2.1.3.3.11 BrowserInfo.Location Property

Returns the URL of the current application.

C#
public string Location;

10.2.1.3.3.12 BrowserInfo.CustomData Property

Gets or sets custom application data.

C#
public string CustomData;
10.2.1.3.3.13 BrowserInfo.ScreenResolution Property

Returns the application screen resolution defined in the application profile.

C#
public int ScreenResolution;

Group
BrowserInfo Properties

10.2.1.3.3.14 BrowserInfo.Orientation Property

Returns the browser's orientation.

C#
public BrowserOrientation Orientation;

Group
BrowserInfo Properties

10.2.1.3.3.15 BrowserInfo.SelectedRule Property

Returns the selected Browser Rule.

C#
public string SelectedRule;

Group
BrowserInfo Properties

10.2.1.4 HTMLDoc Class

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

Class Hierarchy
HTMLDoc
    IHTMLDoc
    IDisposable

C#
public class HTMLDoc : IHTMLDoc, IDisposable;

File
Thinfinity.VirtualUI.cs

© 2018, Cybele Software, Inc.
Namespace

Cybele.Thinfinity Namespace

10.2.1.4.1 HTMLDoc.HTMLDoc Constructor

**C#**

```csharp
public HTMLDoc(IVirtualUI virtualUI);
```

**Description**

This is HTMLDoc, a member of class HTMLDoc.

**Class**

**HTMLDoc Class**

10.2.1.4.2 HTMLDoc Methods

The methods of the HTMLDoc class are listed here.

**Class**

**HTMLDoc Class**

**Public Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateComponent</td>
<td>This is the overview for the CreateComponent method overload.</td>
</tr>
<tr>
<td>GetSafeUrl</td>
<td>This is the overview for the GetSafeUrl method overload.</td>
</tr>
<tr>
<td>ImportHTML</td>
<td>This is the overview for the ImportHTML method overload.</td>
</tr>
<tr>
<td>LoadScript</td>
<td>This is the overview for the LoadScript method overload.</td>
</tr>
<tr>
<td>Dispose</td>
<td>This is Dispose, a member of class HTMLDoc.</td>
</tr>
<tr>
<td>CreateSessionURL</td>
<td>Creates an url pointing to a local filename. This url is valid during the session lifetime and its private to this session.</td>
</tr>
</tbody>
</table>

10.2.1.4.2.1 CreateComponent Method

This is the overview for the CreateComponent method overload.

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTMLDoc.CreateComponent</td>
<td>This is CreateComponent, a member of class HTMLDoc.</td>
</tr>
</tbody>
</table>
**HTMLDoc Methods**

Inserts an HTML. Used to insert regular HTML elements or WebComponents with custom elements.

**C#**

public void CreateComponent(string Id, string Html, IntPtr ReplaceWnd);

**Group**

CreateComponent Method

**C#**

public void CreateComponent(string Id, string Html, Int64 ReplaceWnd);

**Description**

This is CreateComponent, a member of class HTMLDoc.

**Group**

CreateComponent Method

10.2.1.4.2.2 GetSafeUrl Method

This is the overview for the GetSafeUrl method overload.

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTMLDoc.GetSafeUrl (string)</td>
<td>Gets a safe, temporary and unique URL to access any file in the disk.</td>
</tr>
<tr>
<td>HTMLDoc.GetSafeUrl (string, int)</td>
<td>Gets a safe, temporary and unique URL to access any file in the disk.</td>
</tr>
</tbody>
</table>

**Group**

HTMLDoc Methods

Gets a safe, temporary and unique URL to access any file in the disk.

**C#**

public string GetSafeUrl(string filename);
GetSafeUrl Method

Gets a safe, temporary and unique URL to access any file in the disk.

C#  
public string GetSafeUrl(string filename, int minutes);

Group
GetSafeUrl Method

10.2.1.4.2.3 ImportHTML Method

This is the overview for the ImportHTML method overload.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTMLDoc.ImportHTML (string)</td>
<td>Imports an HTML from URL. If Filename is specified, creates a session URL first and then imports the html file from that Filename.</td>
</tr>
<tr>
<td>HTMLDoc.ImportHTML (string, string)</td>
<td>Imports an HTML file from disk and assigns a session url.</td>
</tr>
</tbody>
</table>

Group
HTMLDoc Methods

Imports an HTML from URL. If Filename is specified, creates a session URL first and then imports the html file from that Filename.

C#  
public void ImportHTML(string url);

Group
ImportHTML Method

Imports an HTML file from disk and assigns a session url.

C#  
public void ImportHTML(string url, string filename);
10.2.1.4.2.4 LoadScript Method

This is the overview for the LoadScript method overload.

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTMLDoc.LoadScript (string)</td>
<td>Loads a script from url.</td>
</tr>
<tr>
<td>HTMLDoc.LoadScript (string, string)</td>
<td>Loads a script from URL. If Filename is specified, creates a session URL first and then load the script from that Filename.</td>
</tr>
</tbody>
</table>

### Group

**HTMLDoc Methods**

Loads a script from url.

**C#**

```csharp
public void LoadScript(string url);
```

### Group

**LoadScript Method**

Loads a script from URL. If Filename is specified, creates a session URL first and then load the script from that Filename.

**C#**

```csharp
public void LoadScript(string url, string filename);
```

### Group

**LoadScript Method**

10.2.1.4.2.5 HTMLDoc.Dispose Method

**C#**

```csharp
public void Dispose();
```

### Description

This is Dispose, a member of class HTMLDoc.

### Group

**HTMLDoc Methods**
10.2.1.4.2.6 HTMLDoc.CreateSessionURL Method

Creates an url pointing to a local filename. This url is valid during the session lifetime and its private to this session.

**C#**

```csharp
public void CreateSessionURL(string url, string filename);
```

**Group**

HTMLDoc Methods

10.2.1.5 DevServer Class

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

**Class Hierarchy**

```
DevServer
    + Cybele.Thinfinity.DevServer
    + IDisposable
```

**C#**

```csharp
public class DevServer : IDevServer, IDisposable;
```

**File**

Thinfinity.VirtualUI.cs

**Namespace**

Cybele.Thinfinity Namespace

10.2.1.5.1 DevServer.DevServer Constructor

**C#**

```csharp
public DevServer(IVirtualUI virtualUI);
```

**Description**

This is DevServer, a member of class DevServer.

**Class**

DevServer Class

10.2.1.5.2 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>

10.2.1.5.2.1 DevServer.Dispose Method

C#

```csharp
public void Dispose();
```

Description
This is Dispose, a member of class DevServer.

Group
DevServer Methods

10.2.1.5.3 DevServer Properties

The properties of the DevServer class are listed here.

Class
DevServer 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>

10.2.1.5.3.1 DevServer.Enabled Property

Enables/disables the Development Server.

C#

```csharp
public bool Enabled;
```

Group
**DevServer Properties**

10.2.1.5.3.2 DevServer.Port Property

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

```csharp
public int Port;
```

**Group**

**DevServer Properties**

10.2.1.5.3.3 DevServer.StartBrowser Property

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

```csharp
public bool StartBrowser;
```

**Group**

**DevServer Properties**

10.2.1.6 VirtualUI Class

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

**Class Hierarchy**

```
public class VirtualUI : VirtualUILibrary, IDisposable;
```

**File**

Thinfinity.VirtualUI.cs

**Namespace**

Cybele.Thinfinity Namespace

10.2.1.6.1 VirtualUI.VirtualUI Constructor

```csharp
public VirtualUI();
```
### Description

This is VirtualUI, a member of class VirtualUI.

### Class

**VirtualUI Class**

#### 10.2.1.6.2 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>
<tr>
<td>UploadFile</td>
<td>This is the overview for the UploadFile method overload.</td>
</tr>
<tr>
<td>Dispose</td>
<td>This is Dispose, a member of class VirtualUI.</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>PreviewPdf</td>
<td>Sends the specified PDF file to be shown on the end-user's web browser. It's similar to PrintPdf, except that disables the printing options in the browser. Built-in browser printing commands will be available.</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>TakeScreenshot</td>
<td>Takes a screenshot of a Window.</td>
</tr>
<tr>
<td>ShowVirtualKeyboard</td>
<td>In mobile, shows the keyboard.</td>
</tr>
</tbody>
</table>

#### 10.2.1.6.2.1 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>VirtualUI.DownloadFile (string)</td>
<td>Sends the specified file to the end-user's web browser for saving it in the remote machine.</td>
</tr>
<tr>
<td>VirtualUI.DownloadFile (string, string)</td>
<td>Sends the specified file to the end-user's web browser for saving it in the remote machine.</td>
</tr>
<tr>
<td>VirtualUI.DownloadFile (string, string, string)</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

**VirtualUI Methods**

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

### C#

```csharp
public void DownloadFile(string LocalFilename);
```

## Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string LocalFilename</td>
<td>Name of both the local and remote file.</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#

```csharp
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#

```csharp
public void DownloadFile(string LocalFilename, string RemoteFilename, string MimeType);
```

## Parameters
### Group

**DownloadFile Method**

10.2.1.6.2.2 **Start Method**

This is the overview for the Start method overload.

<table>
<thead>
<tr>
<th>Overload List</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>VirtualUI.Start ()</td>
</tr>
<tr>
<td>VirtualUI.Start (int)</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#

```csharp
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#

```csharp
public bool Start(int timeout);
```

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td>Timeout</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 Timeout parameter.

Group

Start Method

10.2.1.6.2.3 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>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

C#

```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.

C#

```csharp
public void UploadFile();
```
### 10.2.1.6.2.4 VirtualUI.Dispose Method

**C#**
```csharp
public void Dispose();
```

**Description**
This is Dispose, a member of class VirtualUI.

**Group**
VirtualUI Methods

### 10.2.1.6.2.5 VirtualUI.Stop Method

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

**C#**
```csharp
public void Stop();
```

**Group**
VirtualUI Methods

### 10.2.1.6.2.6 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);
```

**Parameters**

<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

### 10.2.1.6.2.7 VirtualUI.PreviewPdf Method

Sends the specified PDF file to be shown on the end-user's web browser. It's similar to PrintPdf, except that disables the printing options in the browser. Built-in browser printing commands will be available.

**C#**
```csharp
```
public void PreviewPdf(string FileName);

## Parameters

<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>

## Group

VirtualUI Methods

### 10.2.1.6.2.8 VirtualUI.OpenLinkDlg Method

Displays a popup with a button to open a web link.

```csharp
public void OpenLinkDlg(string Url, string Caption);
```

## Parameters

<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

### 10.2.1.6.2.9 VirtualUI.SendMessage Method

Sends a data string to the web browser.

```csharp
public void SendMessage(string Data);
```

## Remarks

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

## Group

VirtualUI Methods

### 10.2.1.6.2.10 VirtualUI.AllowExecute Method

Allows the execution of the passed application.

```csharp
public void AllowExecute(string Filename);
```
### Parameters

<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

#### 10.2.1.6.2.11 VirtualUI.SetImageQualityByWnd Method

Allows to the the image quality for the specified window.

**C#**

```csharp
public void SetImageQualityByWnd(long Wnd, string Classname, int Quality);
```

### Parameters

<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

#### 10.2.1.6.2.12 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>
VirtualUI Methods

10.2.1.6.2.13 VirtualUI.ShowVirtualKeyboard Method

In mobile, shows the keyboard.

**C#**

```csharp
public void ShowVirtualKeyboard();
```

---

VirtualUI Methods

10.2.1.6.3 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>Options</td>
<td>Gets/sets the option flags</td>
</tr>
<tr>
<td>BrowserInfo</td>
<td>Contains information regarding the end-user's environment.</td>
</tr>
<tr>
<td>HTMLDoc</td>
<td>Contains methods to modify the behavior on the HTML page.</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>Recorder</td>
<td>Session recording and playback.</td>
</tr>
<tr>
<td>FileSystemFilter</td>
<td>This is FileSystemFilter, a member of class VirtualUI.</td>
</tr>
<tr>
<td>RegistryFilter</td>
<td>This is RegistryFilter, a member of class VirtualUI.</td>
</tr>
</tbody>
</table>
10.2.1.6.3.1 VirtualUI.Active Property

Returns the VirtualUI’s state.

**C#**

```csharp
public bool Active;
```

**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.

10.2.1.6.3.2 VirtualUI.Enabled Property

Enables/disables VirtualUI for the container application.

**C#**

```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.

10.2.1.6.3.3 VirtualUI.DevMode Property

Gets/sets the development mode.

**C#**

```csharp
public bool DevMode;
```

**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.

10.2.1.6.3.4 VirtualUI.StdDialogs Property

Enables/disables the use of standard dialogs.
10.2.1.6.3.5 VirtualUI.Options Property

Gets/sets the option flags

C#
public uint Options;

10.2.1.6.3.6 VirtualUI.BrowserInfo Property

Contains information regarding the end-user's environment.

C#
public BrowserInfo BrowserInfo;

10.2.1.6.3.7 VirtualUI.HTMLDoc Property

Contains methods to modify the behavior on the HTML page.

C#
public HTMLDoc HTMLDoc;

10.2.1.6.3.8 VirtualUI.DevServer Property

Allows for managing the Development Server.

C#
public DevServer DevServer;
10.2.1.6.3.9 VirtualUI.ClientSettings Property

Controls some working parameters on the client side.

C#
public IClientSettings ClientSettings;

Remarks
To record a session:
- Set the Recorder.Filename property with the path and name of the session file to be used. No extension is needed; the recorder will generate two files, with extensions idx and dat.
- Call Recorder.Play, passing a track name. This name will allow to play different recordings in the same session.

To stop recording:
- Call Recorder.Stop.

To play a recorded session:
- Set the Filename property with the path and name of the session file to be played (the file with idx extension).
- Call Play with the range of entries to play.

To play an entire session, pass 0 and the Count property.

To play only a specific track, pass the Position of track to reproduce as From, and the Position of next track as To. For the last track, the To parameter must be the Count property of recorder.

Group
VirtualUI Properties

10.2.1.6.3.10 VirtualUI.Recorder Property

Session recording and playback.

C#
public IRecorder Recorder;

Remarks
To record a session:
- Set the Recorder.Filename property with the path and name of the session file to be used. No extension is needed; the recorder will generate two files, with extensions idx and dat.
- Call Recorder.Play, passing a track name. This name will allow to play different recordings in the same session.

To stop recording:
- Call Recorder.Stop.

To play a recorded session:
- Set the Filename property with the path and name of the session file to be played (the file with idx extension).
- Call Play with the range of entries to play.

To play an entire session, pass 0 and the Count property.

To play only a specific track, pass the Position of track to reproduce as From, and the Position of next track as To. For the last track, the To parameter must be the Count property of recorder.

Group
VirtualUI Properties

10.2.1.6.3.11 VirtualUI.FileSystemFilter Property

C#
public IFileSystemFilter FileSystemFilter;

Description
This is FileSystemFilter, a member of class VirtualUI.
10.2.1.6.3.12 VirtualUI.RegistryFilter Property

**C#**

```csharp
public IRegistryFilter RegistryFilter;
```

**Description**

This is RegistryFilter, a member of class VirtualUI.

10.2.1.6.4 VirtualUI Events

The events of the VirtualUI class are listed here.

**Class**

VirtualUI Class

**Public Events**

<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>
<tr>
<td>OnRecorderChanged</td>
<td>This is OnRecorderChanged, a member of class VirtualUI.</td>
</tr>
</tbody>
</table>

10.2.1.6.4.1 VirtualUI.OnBrowserResize Event

**C#**

```csharp
public event EventHandler<BrowserResizeEventArgs> OnBrowserResize;
```

**Description**

This is OnBrowserResize, a member of class VirtualUI.
10.2.1.6.4.2 VirtualUI.OnGetUploadDir Event

**C#**

```csharp
public event EventHandler<GetUploadDirEventArgs> OnGetUploadDir;
```

**Description**

This is OnGetUploadDir, a member of class VirtualUI.

10.2.1.6.4.3 VirtualUI.OnClose Event

**C#**

```csharp
public event EventHandler<CloseArgs> OnClose;
```

**Description**

This is OnClose, a member of class VirtualUI.

10.2.1.6.4.4 VirtualUI.OnReceiveMessage Event

**C#**

```csharp
public event EventHandler<ReceiveMessageArgs> OnReceiveMessage;
```

**Description**

This is OnReceiveMessage, a member of class VirtualUI.

10.2.1.6.4.5 VirtualUI.OnDownloadEnd Event

**C#**

```csharp
public event EventHandler<DownloadEndArgs> OnDownloadEnd;
```

**Description**
This is OnDownloadEnd, a member of class VirtualUI.

== Group
VirtualUI Events

10.2.1.6.4.6 VirtualUI.OnRecorderChanged Event

C#

```csharp
public event EventHandler<RecorderChangedEventArgs> OnRecorderChanged;
```

== Description
This is OnRecorderChanged, a member of class VirtualUI.

== Group
VirtualUI Events

10.2.1.7 JSObject Class

Represents a custom remotable object.

== Class Hierarchy

```
VirtualUILibrary
   JSObject
      Cybele.Thinity.JSObject
      IDisposable
```

C#

```csharp
public class JSObject : VirtualUILibrary, IJSObject, IDisposable;
```

== File
Thinfinity.VirtualUI.cs

== 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**

### 10.2.1.7.1 JSObject.JSObject Constructor

**C#**

```csharp
public JSObject(string Id);
```

**Description**

This is JSObject, a member of class JSObject.

### 10.2.1.7.2 JSObject Methods

The methods of the JSObject class are listed here.

**Class**

**JSObject Class**

### 10.2.1.7.2.1 JSObject.Dispose Method

**C#**

```csharp
public void Dispose();
```

**Description**

This is Dispose, a member of class JSObject.

**Group**
JSObject Methods

10.2.1.7.2.2 JSObject.FireEvent Method

C#
public void FireEvent(string Name, IJSArguments Arguments);

- Description
This is FireEvent, a member of class JSObject.

- Group
JSObject Methods

10.2.1.7.2.3 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.

C#
public void ApplyChanges();

- Group
JSObject Methods

10.2.1.7.2.4 JSObject.ApplyModel Method

Propagates the whole JSObject definition to the javascript client.

C#
public void ApplyModel();

- Group
JSObject Methods

10.2.1.7.3 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>
</table>

© 2018, Cybele Software, Inc.
### 10.2.1.7.3.1 JSObject.Id Property

Identifier of the object. It must be unique among siblings objects.

**C#**

```csharp
public string Id;
```

≡ Group

**JSObject Properties**

### 10.2.1.7.3.2 JSObject.Properties Property

List containing all properties of this object.

**C#**

```csharp
public IJSProperties Properties;
```

≡ Group

**JSObject Properties**

### 10.2.1.7.3.3 JSObject.Methods Property

List containing all methods of this object.

**C#**

```csharp
public IJSMethods Methods;
```

≡ Group

**JSObject Properties**

### 10.2.1.7.3.4 JSObject.Events Property

List containing all events of this object.

**C#**

```csharp
public IJSEvents Events;
```
10.2.1.7.3.5 JSObject.Objects Property

List containing all events of this object.

C#

```csharp
public IJSObjects Objects;
```

10.2.1.7.4 JSObject Events

The events of the JSObject class are listed here.

10.2.1.7.4.1 JSObject.OnExecuteMethod Event

Fired when a method is executed on the remote object.

C#

```csharp
public event EventHandler<JSExecuteMethodEventArgs> OnExecuteMethod;
```

10.2.1.7.4.2 JSObject.OnPropertyChanged Event

Fired when a property value has been changed on the remote object.

C#

```csharp
public event EventHandler<JSPropertyChangeEventArgs> OnPropertyChanged;
```
10.2.1.8 JSBinding Class

C# [ComVisible(true), ClassInterface(ClassInterfaceType.AutoDual), ComDefaultInterface(typeof(IJSBinding))]
public class JSBinding : IJSBinding, IDisposable;

File
Thinfinity.VirtualUI.cs

Description
This is class Cybele.Thinfinity.JSBinding.

Namespace
Cybele.Thinfinity Namespace

10.2.1.8.1 JSBinding.JSBinding Constructor

C#
public JSBinding(JSPropertySet Proc);

Description
This is JSBinding, a member of class JSBinding.

Class
JSBinding Class

10.2.1.8.2 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>

10.2.1.8.2.1 JSBinding.Dispose Method

_C#

```csharp
public void Dispose();
```

_Description_

This is Dispose, a member of class JSBinding.

_Group_

JSBinding Methods

10.2.1.8.2.2 JSBinding.Set Method

_C#

```csharp
public void Set(IJSObject Parent, IJSProperty Prop);
```

_Description_

This is Set, a member of class JSBinding.

_Group_

JSBinding Methods

10.2.1.9 JSCallback Class

_Class Hierarchy_

- JSCallback
- Cybele.Thinfinity.JSCallback
- IDisposable

_C#

```csharp
[ComVisible(true), ClassInterface(ClassInterfaceType.AutoDual), ComDefaultInterface(typeof(IJSCallback))]
public class JSCallback : IJSCallback, IDisposable;
```

_File_

Thinfinity.VirtualUI.cs

_Description_
This is class Cybele.Thinfinity.JSCallback.

## Namespace

**Cybele.Thinfinity Namespace**

### 10.2.1.9.1 JSCallback.JSCallback Constructor

#### C#

```csharp
public JSCallback(JSMethodCallback Proc);
```

#### Description

This is JSCallback, a member of class JSCallback.

### Class

**JSCallback Class**

### 10.2.1.9.2 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>

### 10.2.1.9.2.1 JSCallback.Dispose Method

#### C#

```csharp
public void Dispose();
```

#### Description

This is Dispose, a member of class JSCallback.

### Group

**JSCallback Methods**
10.2.1.9.2.2 JSCallback.Callback Method

**C#**

```csharp
public void Callback(IJSObject Parent, IJSMethod Method);
```

**Description**

This is Callback, a member of class JSCallback.

**Group**

JSCallback Methods

10.2.2 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

10.2.2.1 Server Class

Main class. Contains methods and properties to manage all Server configuration.

**Class Hierarchy**
C#  
**public class Server :** VirtualUISLibrary, IServer, IDisposable;  

**File**  
Thinfinity.VirtualUI.Settings.cs  

**Namespace**  
Cybele.Thinfinity.Settings.VirtualUI Namespace  

10.2.2.1.1 Server.Server Constructor  

C#  
```csharp  
public Server();  
```  

**Description**  
This is Server, a member of class Server.  

**Class**  
Server Class  

10.2.2.1.2 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>
<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>
<tr>
<td>ShowSection</td>
<td>Makes visible a configuration section in the VirtualUI Server Manager GUI.</td>
</tr>
</tbody>
</table>
10.2.2.1.2.1 Server.Dispose Method

```csharp
public void Dispose();
```

**Description**
This is Dispose, a member of class Server.

**Group**
Server Methods

10.2.2.1.2.2 Server.Load Method

Loads all the configuration entries and profiles from file. It's automatically called by constructor.

```csharp
public void Load();
```

**Group**
Server Methods

10.2.2.1.2.3 Server.Save Method

Saves the entire configuration parameters and profiles.

```csharp
public void Save();
```

**Group**
Server Methods

10.2.2.1.2.4 Server.HideSection Method

Hides a configuration section in the VirtualUI Server Manager GUI.

```csharp
public void HideSection(ServerSection section);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerSection section</td>
<td>The Server configuration section to hide to user. Use one of the following constants:</td>
</tr>
<tr>
<td></td>
<td>- SRVSECGENERAL: Hides the General tab, that contains the Binding configuration.</td>
</tr>
<tr>
<td></td>
<td>- SRVSEC_RDS: Hides the tab with the Remote Desktop Services account configuration.</td>
</tr>
<tr>
<td></td>
<td>- SRVSEC_APPLICATIONS: Hides the list of</td>
</tr>
</tbody>
</table>
Group

Server Methods

10.2.2.1.2.5 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>
<tbody>
<tr>
<td>ServerSection section</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>• SRVSECAPPLICATIONS: Shows the list of applications.</td>
</tr>
<tr>
<td></td>
<td>• SRVSEC_LICENSES: Shows the tab with License information.</td>
</tr>
</tbody>
</table>

Group

Server Methods

10.2.2.1.3 Server Properties

The properties of the Server class are listed here.

Class

Server Class

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![R] Binding</td>
<td>Returns the Server's Binding configuration.</td>
</tr>
<tr>
<td>![R] Certificate</td>
<td>Returns the Server's certificate configuration for SSL protocol.</td>
</tr>
<tr>
<td>![R] RDSAccounts</td>
<td>Returns the list of Remote Desktop Services accounts.</td>
</tr>
<tr>
<td>![R] Profiles</td>
<td>Returns the profiles list.</td>
</tr>
</tbody>
</table>
### License

**License**

Returns the current *Server's* licence.

### Gateways

**Gateways**

Returns the current *Server's* gateways.

#### 10.2.2.1.3.1 Server.Binding Property

Returns the *Server's* Binding configuration.

**C#**

```csharp
public IBinding Binding;
```

- **See Also**
  
  IBinding interface

- **Group**
  
  Server Properties

#### 10.2.2.1.3.2 Server.Certificate Property

Returns the *Server's* certificate configuration for SSL protocol.

**C#**

```csharp
public ICertificate Certificate;
```

- **See Also**
  
  ICertificate interface

- **Group**
  
  Server Properties

#### 10.2.2.1.3.3 Server.RDSAccounts Property

Returns the list of Remote Desktop Services accounts.

**C#**

```csharp
public IRDSAccounts RDSAccounts;
```

- **See Also**
  
  IRDSAccounts interface

- **Group**
  
  Server Properties
10.2.2.1.3.4 Server.Profiles Property

Returns the profiles list.

```csharp
public IProfiles Profiles;
```

See Also

IProfiles interface

Group

Server Properties

10.2.2.1.3.5 Server.License Property

Returns the current Server's licence.

```csharp
public ILicense License;
```

See Also

ILicense interface

Group

Server Properties

10.2.2.1.3.6 Server.Gateways Property

Returns the current Server's gateways.

```csharp
public IGateways Gateways;
```

See Also

IGateway interface

Group

Server Properties

10.2.2.2 ServerUtils Class

Helper functions.
Class Hierarchy

C#

```csharp
public class ServerUtils;
```

File
Thinfinity.VirtualUI.Settings.cs

Namespace

Cybele.Thinfinity.Settings.VirtualUI Namespace

10.2.2.2.1 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>

10.2.2.1.1 ServerUtils.RunAsAdmin Method

Runs an application in elevated mode. This mode is required to save the Server's configuration in protected files.

```
C#

public static void RunAsAdmin(string fileName, string parameters);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</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

10.2.2.1.2 ServerUtils.Base64ToIcon Method

Converts the IProfile.IconData (base64 string) to a PNG image.

C#

```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

10.2.2.1.3 ServerUtils.IconToBase64 Method

Converts a PNG image to be stored in IProfile.IconData (as base64 string).

C#

```c#
public static string IconToBase64(Image png);
```

Description

param name="png"> The image to be encoded in base64.

Group
ServerUtils Methods

10.3 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>
10.3.1 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>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>THTMLDoc</td>
<td>Main class. Has methods, properties and events to allow to manage some web behavior.</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

10.3.1.1 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

```
  InterfaceObject
    VirtualUI_SDK.TBrowserInfo
```

Pascal
**TBrowserInfo** = class(TInterfacedObject, IBrowserInfo);

### 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>
<tr>
<td>CustomData</td>
<td>Gets or sets custom user data.</td>
</tr>
<tr>
<td>SelectedRule</td>
<td>Returns the selected Browser Rule.</td>
</tr>
</tbody>
</table>
10.3.1.1.1  TBrowserInfo.ViewWidth Property

Returns the width of the VirtualUI Viewer.

Pascal
property ViewWidth: Integer;

Group
TBrowserInfo Properties

10.3.1.1.2  TBrowserInfo.ViewHeight Property

Returns the height of the VirtualUI Viewer.

Pascal
property ViewHeight: Integer;

Group
TBrowserInfo Properties

10.3.1.1.3  TBrowserInfo.BrowserWidth Property

Returns the width of the HTML element containing the VirtualUI Viewer.

Pascal
property BrowserWidth: Integer;

Group
TBrowserInfo Properties

10.3.1.1.4  TBrowserInfo.BrowserHeight Property

Returns the height of the HTML element containing the VirtualUI Viewer.

Pascal
property BrowserHeight: Integer;

Group
TBrowserInfo Properties

10.3.1.1.5  TBrowserInfo.ScreenWidth Property

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

Pascal
property ScreenWidth: Integer;
10.3.1.1.1.6  TBrowserInfo.ScreenHeight Property

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

**Pascal**

```pascal
property ScreenHeight: Integer;
```

10.3.1.1.1.7  TBrowserInfo.Username Property

Returns the logged-on Username.

**Pascal**

```pascal
property Username: WideString;
```

10.3.1.1.1.8  TBrowserInfo.IPAddress Property

Returns the client's IP address.

**Pascal**

```pascal
property IPAddress: WideString;
```

10.3.1.1.1.9  TBrowserInfo.UserAgent Property

Returns the browser's User Agent string.

**Pascal**

```pascal
property UserAgent: WideString;
```
10.3.1.1.1.10 TBrowserInfo.ScreenResolution Property

Returns the application screen resolution defined in the application profile.

```pascal
property ScreenResolution: Integer;
```

10.3.1.1.1.11 TBrowserInfo.Orientation Property

Returns the browser's orientation.

```pascal
property Orientation: Orientation;
```

10.3.1.1.1.12 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;
```

10.3.1.1.1.13 TBrowserInfo.Location Property

Returns the URL of the current application.

```pascal
property Location: WideString;
```

10.3.1.1.1.14 TBrowserInfo.CustomData Property

Gets or sets custom user data.

```pascal
property CustomData: WideString;
```
10.3.1.1.15  TBrowserInfo.SelectedRule Property

Returns the selected Browser Rule.

Pascal

property SelectedRule: WideString;

Group

TBrowserInfo Properties

10.3.1.2  TClientSettings Class

Allows to set some client settings.

Class Hierarchy

Pascal

TClientSettings = class(TInterfacedObject, IClientSettings);

File

VirtualUI_SDK

Namespace

VirtualUI_SDK Namespace

10.3.1.2.1  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</td>
</tr>
</tbody>
</table>
10.3.1.2.1.1  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;

10.3.1.2.1.2  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;

10.3.1.2.1.3  TClientSettings.CursorVisible Property

Hides/shows the mouse pointer.

Pascal

property CursorVisible: WordBool;

10.3.1.3  TDevServer Class

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

Class Hierarchy
Symbol Reference 449

Pascal
TDevServer = class(TInterfacedObject, IDevServer);

File
VirtualUI_SDK

Namespace
VirtualUI_SDK Namespace

10.3.1.3.1  TDevServer.Create Constructor

Pascal
constructor Create(AVirtualUI: IVirtualUI);

Description
This is Create, a member of class TDevServer.

Class
TDevServer Class

10.3.1.3.2  TDevServer.Destroy Destructor

Pascal
destructor Destroy; override;

Description
This is Destroy, a member of class TDevServer.

Class
TDevServer Class

10.3.1.3.3  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>

### 10.3.1.3.3.1 TDevServer.Enabled Property

Enables/disables the Development Server.

**Pascal**

```pascal
property Enabled: WordBool;
```

**Group**

TDevServer Properties

### 10.3.1.3.3.2 TDevServer.Port Property

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

**Pascal**

```pascal
property Port: Integer;
```

**Group**

TDevServer Properties

### 10.3.1.3.3.3 TDevServer.StartBrowser Property

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

**Pascal**

```pascal
property StartBrowser: WordBool;
```

**Group**

TDevServer Properties

### 10.3.1.4 TJSBinding Class

**Class Hierarchy**

```pascal
TJSBinding = class(TInterfacedObject, IJSBinding);
```

© 2018, Cybele Software, Inc.
File
VirtualUI_SDK

Description
This is class VirtualUI_SDK.TJSBinding.

Namespace
VirtualUI_SDK Namespace

10.3.1.5 TJSCallback Class

Class Hierarchy
- TInterfacedObject
- VirtualUI_SDK.TJSCallback
- TJSCallback

Pascal
TJSCallback = class(TInterfacedObject, IJSCallback);

File
VirtualUI_SDK

Description
This is class VirtualUI_SDK.TJSCallback.

Namespace
VirtualUI_SDK Namespace

10.3.1.6 TJSObject Class

Represents a custom remotable object.

Class Hierarchy
- TInterfacedObject
- VirtualUI_SDK.TJSObject
- TJSObject

Pascal
TJSObject = class(TInterfacedObject, IJSObject);

File
 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 (IJSMeth) 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

10.3.1.6.1 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>

10.3.1.6.1.1 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>
10.3.1.6.1.2 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;

10.3.1.6.1.3 TJSObject.ApplyModel Method

Propagates the whole JSObject definition to the javascript client.

Pascal

procedure ApplyModel; safecall;

10.3.1.6.2 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>
10.3.1.6.2.1  TJSObject.Id Property

Identifier of the object. It must be unique among siblings objects.

**Pascal**

```pascal
property Id: WideString;
```

**Group**

**TJSObject Properties**

10.3.1.6.2.2  TJSObject.Properties Property

List containing all properties of this object.

**Pascal**

```pascal
property Properties: IJSProperties;
```

**Group**

**TJSObject Properties**

10.3.1.6.2.3  TJSObject.Methods Property

List containing all methods of this object.

**Pascal**

```pascal
property Methods: IJSMethods;
```

**Group**

**TJSObject Properties**

10.3.1.6.2.4  TJSObject.Events Property

List containing all events of this object.

**Pascal**

```pascal
property Events: IJSEvents;
```

**Group**

**TJSObject Properties**

10.3.1.6.2.5  TJSObject.Objects Property

List containing all events of this object.

**Pascal**

```pascal
property Objects: IJSObjects;
```
10.3.1.6.3  TJSObject Events

The events of the TJSObject class are listed here.

### 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>

#### 10.3.1.6.3.1 TJSObject.OnExecuteMethod Event

Fired when a method is executed on the remote object.

**Pascal**

```pascal
property OnExecuteMethod: TExecuteMethodEvent;
```

#### 10.3.1.6.3.2 TJSObject.OnPropertyChange Event

Fired when a property value has been changed on the remote object.

**Pascal**

```pascal
property OnPropertyChange: TPropertyChangeEvent;
```

### 10.3.1.7 THTHMDoc Class

Main class. Has methods, properties and events to allow to manage some web behavior.
```
Pascal
THTMLDoc = class(TInterfacedObject, IHTMLDoc);
```

## File
VirtualUI_SDK

## Namespace
VirtualUI_SDK Namespace

### 10.3.1.7.1 THTMLDoc Methods

The methods of the THTMLDoc class are listed here.

#### Class

THTMLDoc Class

### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateSessionURL</td>
<td>Creates an url pointing to a local filename. This url is valid during the session lifetime and its private to this session.</td>
</tr>
<tr>
<td>CreateComponent</td>
<td>Inserts an HTML. Used to insert regular HTML elements or WebComponents with custom elements.</td>
</tr>
<tr>
<td>GetSafeURL</td>
<td>Returns a safe, temporary and unique URL to access any local file.</td>
</tr>
<tr>
<td>LoadScript</td>
<td>Loads a script from URL. If Filename is specified, creates a session URL first and then load the script from that Filename.</td>
</tr>
<tr>
<td>ImportHTML</td>
<td>Imports an HTML from URL. If Filename is specified, creates a session URL first and then imports the html file from that Filename.</td>
</tr>
</tbody>
</table>

#### 10.3.1.7.1.1 THTMLDoc.CreateSessionURL Method

Creates an url pointing to a local filename. This url is valid during the session lifetime and its private to this session.

```
Pascal
procedure CreateSessionURL(const Url: WideString; const Filename: WideString); safecall;
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Url: WideString</td>
<td>Arbitary relative url.</td>
</tr>
</tbody>
</table>

© 2018, Cybele Software, Inc.
10.3.1.7.1.2  THTMLDoc.CreateComponent Method

Inserts an HTML. Used to insert regular HTML elements or WebComponents with custom elements.

**Pascal**

```pascal
procedure CreateComponent(const Id: WideString; const Html: WideString; ReplaceWnd: Int64); safecall;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Id: WideString</td>
<td>ID to assign to the main element of the HTML to be inserted</td>
</tr>
<tr>
<td>const Html: WideString</td>
<td>HTML snippet</td>
</tr>
<tr>
<td>ReplaceWnd: Int64</td>
<td>Wnd to be replaced and tied to</td>
</tr>
</tbody>
</table>

**Remarks**

When ReplaceWnd is <> 0 and points to a valid window handle, the positioning of the main element will follow the Wnd positioning, simulating an embedding.

10.3.1.7.1.3  THTMLDoc.GetSafeURL Method

Returns a safe, temporary and unique URL to access any local file.

**Pascal**

```pascal
function GetSafeURL(const Filename: WideString; Minutes: Integer): WideString; safecall;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Filename: WideString</td>
<td>Local filename</td>
</tr>
<tr>
<td>Minutes: Integer</td>
<td>Expiration in minutes</td>
</tr>
</tbody>
</table>

© 2018, Cybele Software, Inc.
10.3.1.7.1.4 THTMLDoc.LoadScript Method

Loads a script from URL. If Filename is specified, creates a session URL first and then load the script from that Filename.

**Pascal**

```pascal
procedure LoadScript(const Url: WideString; const Filename: WideString = ''); safecall; overload;
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Url: WideString</td>
<td>relative URL</td>
</tr>
<tr>
<td>const Filename: WideString = ''</td>
<td>Local filename (optional)</td>
</tr>
</tbody>
</table>

**Group**

THTMLDoc Methods

10.3.1.7.1.5 THTMLDoc.ImportHTML Method

Imports an HTML from URL. If Filename is specified, creates a session URL first and then imports the html file from that Filename.

**Pascal**

```pascal
procedure ImportHTML(const Url: WideString; const Filename: WideString = ''); safecall; overload;
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const Url: WideString</td>
<td>relative URL</td>
</tr>
<tr>
<td>const Filename: WideString = ''</td>
<td>Local filename (optional)</td>
</tr>
</tbody>
</table>

**Group**

THTMLDoc Methods

10.3.1.8 TVirtualUI Class

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

**Class Hierarchy**

```
TInterfacedObject
 |   TVirtualUI = class(TInterfacedObject, IVirtualUI);
```

**File**

© 2018, Cybele Software, Inc.
VirtualUI_SDK

Namespace
VirtualUI_SDK Namespace

10.3.1.8.1 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>PreviewPdf</td>
<td>Sends the specified PDF file to be shown on the end-user's web browser. It's similar to PrintPdf, except that disables the printing options in the browser. Built-in browser printing commands will be available.</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>

10.3.1.8.1.1 DownloadFile Method

This is the overview for the DownloadFile method overload.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</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; 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>

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);
```

### 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);
```

### 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>
</tbody>
</table>

© 2018, Cybele Software, Inc.
### 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**

10.3.1.8.1.2 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

function Start: WordBool; overload;

Group

Start Method

10.3.1.8.1.3 UploadFile Method

This is the overview for the UploadFile method overload.

<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

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

procedure UploadFile; overload;

Group

UploadFile Method

10.3.1.8.1.4 TVirtualUI.Stop Method

Deactivates VirtualUI, closing the connection with the end-user's web browser.
Symbol Reference 463

Pascal
procedure Stop; safecall;

Group
TVirtualUI Methods

10.3.1.8.1.5  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

10.3.1.8.1.6  TVirtualUI.PreviewPdf Method

Sends the specified PDF file to be shown on the end-user's web browser. It's similar to PrintPdf, except that disables the printing options in the browser. Built-in browser printing commands will be available.

Pascal
procedure PreviewPdf(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>

Group
TVirtualUI Methods

10.3.1.8.1.7  TVirtualUI.TakeScreenshot Method

Takes a screenshot of a Window.

Pascal
function TakeScreenshot(Wnd: Integer; const FileName: WideString): WordBool; safecall;

© 2018, Cybele Software, Inc.
### 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**

#### 10.3.1.8.1.8  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;
```

<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>

#### Group

**TVirtualUI Methods**

#### 10.3.1.8.1.9  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.

#### Group

**TVirtualUI Methods**

#### 10.3.1.8.1.10  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

10.3.1.8.1.11 TVirtualUI.SetImageQualityByWnd Method

Allows to the the image quality for the specified window.

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

10.3.1.8.1.12 TVirtualUI.ShowVirtualKeyboard Method

In mobile, shows the keyboard.

Pascal

procedure ShowVirtualKeyboard; safecall;

Group

TVirtualUI Methods

10.3.1.8.2 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 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>
<tr>
<td>Recorder</td>
<td>Session recording and playback.</td>
</tr>
<tr>
<td>FileSystemFilter</td>
<td>FileSystem virtualization</td>
</tr>
<tr>
<td>RegistryFilter</td>
<td>Registry virtualization</td>
</tr>
<tr>
<td>Options</td>
<td>Option flags.</td>
</tr>
<tr>
<td>HTMLDoc</td>
<td>Contains methods to modify the behavior on the HTML page.</td>
</tr>
</tbody>
</table>

### 10.3.1.8.2.1 TVirtualUI.Active Property

Returns the VirtualUI’s state.

**Pascal**

```pascal
property Active: WordBool;
```

### Group

**TVirtualUI Properties**

### 10.3.1.8.2.2 TVirtualUI.Enabled Property

Enables/disables VirtualUI for the container application.

**Pascal**

```pascal
property Enabled: WordBool;
```
10.3.1.8.2.3  TVirtualUI.DevMode Property

Gets/sets the development mode.

Pascal

property DevMode: 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.

Group

10.3.1.8.2.4  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

10.3.1.8.2.5  TVirtualUI.BrowserInfo Property

Contains information regarding the end-user's environment.

Pascal

property BrowserInfo: IBrowserInfo;

Group

10.3.1.8.2.6  TVirtualUI.DevServer Property

Allows for managing the Development Server.
property DevServer: IDevServer;

Group

TVirtualUI Properties

10.3.1.8.2.7 TVirtualUI.ClientSettings Property

Controls some working parameters on the client side.

Pascal

property ClientSettings: IClientSettings;

Group

TVirtualUI Properties

10.3.1.8.2.8 TVirtualUI.Recorder Property

Session recording and playback.

Pascal

property Recorder: IRecorder;

Remarks

To record a session:
- Set the Recorder.Filename property with the path and name of the session file to be used. No extension is needed; the recorder will generate two files, with extensions idx and dat.
- Call Recorder.Play, passing a track name. This name will allow to play different recordings in the same session.

To stop recording:
- Call Recorder.Stop.

To play a recorded session:
- Set the Filename property with the path and name of the session file to be played (the file with idx extension).
- Call Play with the range of entries to play.

To play an entire session, pass 0 and the Count property.

To play only a specific track, pass the Position of track to reproduce as From, and the Position of next track as To. For the last track, the To parameter must be the Count property of recorder.
10.3.1.8.2.9 TVirtualUI.FileSystemFilter Property

FileSystem virtualization

Pascal

property FileSystemFilter: IFileSystemFilter;

Group

TVirtualUI Properties

10.3.1.8.2.10 TVirtualUI.RegistryFilter Property

Registry virtualization

Pascal

property RegistryFilter: IRegistryFilter;

Group

TVirtualUI Properties

10.3.1.8.2.11 TVirtualUI.Options Property

Option flags.

Pascal

property Options: LongWord;

Group

TVirtualUI Properties

10.3.1.8.2.12 TVirtualUI.HTMLDoc Property

Contains methods to modify the behavior on the HTML page.

Pascal

property HTMLDoc: IHTMLDoc;

Group

TVirtualUI Properties

10.3.1.8.3 TVirtualUI Events

The events of the TVirtualUI class are listed here.
TVirtualUI Class

<table>
<thead>
<tr>
<th>Public Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>OnGetUploadDir</td>
</tr>
<tr>
<td>OnBrowserResize</td>
</tr>
<tr>
<td>OnReceiveMessage</td>
</tr>
<tr>
<td>OnClose</td>
</tr>
<tr>
<td>OnDownloadEnd</td>
</tr>
<tr>
<td>OnRecorderChanged</td>
</tr>
</tbody>
</table>

10.3.1.8.3.1 TVirtualUI.OnGetUploadDir Event

Fires during an upload request, allowing you to change the save folder.

**Pascal**

```pascal
property OnGetUploadDir: TGetUploadDirEvent;
```

**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.

10.3.1.8.3.2 TVirtualUI.OnBrowserResize Event

Fires when the VirtualUI Viewer's container window resizes. Normally, when the browser resizes.

**Pascal**

```pascal
property OnBrowserResize: TBrowserResizeEvent;
```

**Remarks**

10.3.1.8.3.3 TVirtualUI.OnReceiveMessage Event

Fires when a custom data string is sent from the web browser page.

**Pascal**

```pascal
```
property OnReceiveMessage: TReceiveMessageEvent;

Group
TVirtualUI Events

10.3.1.8.3.4 TVirtualUI.OnClose Event

Fires when the browser window is about to close.

Pascal
property OnClose: TCloseEvent;

Group
TVirtualUI Events

10.3.1.8.3.5 TVirtualUI.OnDownloadEnd Event

Fires when the file has been sent.

Pascal
property OnDownloadEnd: TDownloadEndEvent;

Group
TVirtualUI Events

10.3.1.8.3.6 TVirtualUI.OnRecorderChanged Event

Fires when there is a change in the recording or playback status.

Pascal
property OnRecorderChanged: TRecorderChangedEvent;

Group
TVirtualUI Events

10.3.1.9 VirtualUI_SDK.GetDllDir Function

Returns the path where Thinfinity.VirtualUI.dll is located.

Pascal
function GetDllDir: string;

File
VirtualUI_SDK
Namespace

VirtualUI_SDK Namespace

10.3.1.1 VirtualUI_SDK.VirtualUI Function

Returns a global VirtualUI object.

Pascal

function VirtualUI: TVirtualUI;

File

VirtualUI_SDK

Namespace

VirtualUI_SDK Namespace

10.3.2 VirtualUI_Settings Unit

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoGateways</td>
<td>This is class VirtualUI_Settings.CoGateways.</td>
</tr>
<tr>
<td>TServer</td>
<td>Main class. Contains methods and properties to manage all Server configuration.</td>
</tr>
</tbody>
</table>

Constants

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IID_IGateways</td>
<td>This is constant VirtualUI_Settings.IID_IGateways.</td>
</tr>
<tr>
<td>CLASS_Gateways</td>
<td>This is constant VirtualUI_Settings.CLASS_Gateways.</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
## Interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGateways</td>
<td>This is class VirtualUI_Settings.IGateways.</td>
</tr>
<tr>
<td>IGatewaysDisp</td>
<td>This is class VirtualUI_Settings.IGatewaysDisp.</td>
</tr>
</tbody>
</table>

## Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gateways</td>
<td>This is type VirtualUI_Settings.Gateways.</td>
</tr>
</tbody>
</table>

### 10.3.2.1 VirtualUI_Settings.IID_IGateways Constant

#### Pascal

```
IID_IGateways: TGUID = '{716BBB17-7A57-46D1-93BB-2C8A947E1F6B}';
```

#### File

VirtualUI_Settings

#### Description

This is constant VirtualUI_Settings.IID_IGateways.

#### Namespace

VirtualUI_Settings Unit

### 10.3.2.2 VirtualUI_Settings.CLASS_Gateways Constant

#### Pascal

```
CLASS_Gateways: TGUID = '{F2A03C8E-F1FA-4799-B0BC-CDB40BB4C020}';
```

#### File

VirtualUI_Settings

#### Description

This is constant VirtualUI_Settings.CLASS_Gateways.

#### Namespace

VirtualUI_Settings Unit
10.3.2.3 VirtualUI_Settings.Gateways Type

Pascal
Gateways = IGateways;

File
VirtualUI_Settings

Description
This is type VirtualUI_Settings.Gateways.

Namespace
VirtualUI_Settings Unit

10.3.2.4 IGateways Interface

Class Hierarchy

Pascal
IGateways = interface (IDispatch);

File
VirtualUI_Settings

Description
This is class VirtualUI_Settings.IGateways.

Namespace
VirtualUI_Settings Unit

10.3.2.4.1 IGateways Methods

The methods of the IGateways class are listed here.

Interface
IGateways Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2018, Cybele Software, Inc.
10.3.2.4.1.1 IGateways.Get_Count Method

**Pascal**

`function Get_Count: Integer; safecall;`

**Description**

This is Get_Count, a member of class IGateways.

**Group**

IGateways Methods

10.3.2.4.1.2 IGateways.Get_Item Method

**Pascal**

`function Get_Item(index: Integer): WideString; safecall;`

**Description**

This is Get_Item, a member of class IGateways.

**Group**

IGateways Methods

10.3.2.4.1.3 IGateways.Add Method

**Pascal**

`procedure Add(const URL: WideString); safecall;`

**Description**

This is Add, a member of class IGateways.

**Group**

IGateways Methods
10.3.2.4.1.4 IGateways.Delete Method

**Pascal**

```pascal
procedure Delete(Index: Integer); safecall;
```

**Description**

This is Delete, a member of class IGateways.

**Group**

IGateways Methods

10.3.2.4.2 IGateways Properties

The properties of the IGateways class are listed here.

**Interface**

IGateways Interface

**Public Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![R]</td>
<td>![R]</td>
</tr>
<tr>
<td>Count</td>
<td>This is Count, a member of class IGateways.</td>
</tr>
<tr>
<td>![R]</td>
<td>![R]</td>
</tr>
<tr>
<td>Item</td>
<td>This is Item, a member of class IGateways.</td>
</tr>
</tbody>
</table>

10.3.2.4.2.1 IGateways.Count Property

**Pascal**

```pascal
property Count: Integer;
```

**Description**

This is Count, a member of class IGateways.

**Group**

IGateways Properties

10.3.2.4.2.2 IGateways.Item Property

**Pascal**

```pascal
property Item [index: Integer]: WideString;
```

**Description**

This is Item, a member of class IGateways.
10.3.2.5 IGatewaysDisp Interface

Class Hierarchy

Pascal
IGatewaysDisp = dispinterface;

File
VirtualUI_Settings

Description
This is class VirtualUI_Settings.IGatewaysDisp.

Namespace
VirtualUI_Settings Unit

10.3.2.5.1 IGatewaysDisp Methods

The methods of the IGatewaysDisp class are listed here.

Interface
IGatewaysDisp Interface

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>This is Add, a member of class IGatewaysDisp.</td>
</tr>
<tr>
<td>Delete</td>
<td>This is Delete, a member of class IGatewaysDisp.</td>
</tr>
</tbody>
</table>

10.3.2.5.1.1 IGatewaysDisp.Add Method

Pascal

procedure Add(const URL: WideString);

Description
This is Add, a member of class IGatewaysDisp.
10.3.2.5.2 IGatewaysDisp Properties

The properties of the IGatewaysDisp class are listed here.

10.3.2.5.2.1 IGatewaysDisp.Count Property

Pascal

property Count: Integer;

Description

This is Count, a member of class IGatewaysDisp.

10.3.2.5.2.2 IGatewaysDisp.Item Property

Pascal

property Item [index: Integer]: WideString;
Description
This is Item, a member of class IGatewaysDisp.

Group
IGatewaysDisp Properties

10.3.2.6 CoGateways Class

Class Hierarchy
Pascal
CoGateways = class;

File
VirtualUI_Settings

Description
This is class VirtualUI_Settings.CoGateways.

Namespace
VirtualUI_Settings Unit

10.3.2.6.1 CoGateways Methods
The methods of the CoGateways class are listed here.

Class
CoGateways Class

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>This is Create, a member of class CoGateways.</td>
</tr>
<tr>
<td>CreateRemote</td>
<td>This is CreateRemote, a member of class CoGateways.</td>
</tr>
</tbody>
</table>

10.3.2.6.1.1 CoGateways.Create Method

Pascal

```pascal
class function Create: IGateways;
```
Description
This is Create, a member of class CoGateways.

Group
CoGateways Methods

10.3.2.6.1.2 CoGateways.CreateRemote Method

Pascal
class function CreateRemote(const MachineName: string): IGateways;

Description
This is CreateRemote, a member of class CoGateways.

Group
CoGateways Methods

10.3.2.7 TServer Class

Main class. Contains methods and properties to manage all Server configuration.

Class Hierarchy

Pascal
TServer = class(TInterfacedObject, IServer);

File
VirtualUI_Settings

Namespace
VirtualUI_Settings Unit

10.3.2.7.1 TServer.Create Constructor

Pascal
constructor Create;

Description
This is Create, a member of class TServer.
## TServer Class

### TServer.Destroy Destructor

**Pascal**

destructor Destroy; override;

**Description**

This is Destroy, a member of class TServer.

## TServer Methods

The methods of the TServer class are listed here.

### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Load</strong></td>
<td>Loads all the configuration entries and profiles from file. It is automatically called by constructor.</td>
</tr>
<tr>
<td><strong>Save</strong></td>
<td>Saves the entire configuration parameters and profiles.</td>
</tr>
<tr>
<td><strong>HideSection</strong></td>
<td>Hides a configuration section in the VirtualUI Server Manager GUI.</td>
</tr>
<tr>
<td><strong>ShowSection</strong></td>
<td>Makes visible a configuration section in the VirtualUI Server Manager GUI.</td>
</tr>
</tbody>
</table>

### TServer.Load Method

Loads all the configuration entries and profiles from file. It is automatically called by constructor.

**Pascal**

procedure Load; safecall;
10.3.2.7.3.2 TServer.Save Method

Saves the entire configuration parameters and profiles.

**Pascal**

```pascal
procedure Save; safecall;
```

**Group**

TServer Methods

10.3.2.7.3.3 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>
<tbody>
<tr>
<td>section: ServerSection</td>
<td>The Server configuration section to hide to user. Use one of the following constants:</td>
</tr>
<tr>
<td></td>
<td>• SRVSEC_GENERAL: Hides the General tab, that contains the Binding configuration.</td>
</tr>
<tr>
<td></td>
<td>• SRVSEC_RDS: Hides the tab with the Remote Desktop Services account configuration.</td>
</tr>
<tr>
<td></td>
<td>• SRVSEC_APPLICATIONS: Hides the list of applications.</td>
</tr>
<tr>
<td></td>
<td>• SRVSEC_LICENSES: Hides the tab with License information.</td>
</tr>
</tbody>
</table>

**Description**

**Group**

TServer Methods

10.3.2.7.3.4 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>
</tbody>
</table>
Description

Group

TServer Methods

10.3.2.7.4 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>
<tr>
<td>Gateways</td>
<td>Returns the current Server's gateways.</td>
</tr>
</tbody>
</table>

10.3.2.7.4.1 TServer.Binding Property

Returns the Server's Binding configuration.

Pascal

property Binding: IBinding;

See Also

IBinding interface

Group

TServer Properties
10.3.2.7.4.2  TServer.Certificate Property

Returns the Server’s certificate configuration for SSL protocol.

```pascal
property Certificate: ICertificate;
```

See Also

ICertificate interface

Group

TServer Properties

10.3.2.7.4.3  TServer.RDSAccounts Property

Returns the list of Remote Desktop Services accounts.

```pascal
property RDSAccounts: IRDSAccounts;
```

See Also

IRDSAccounts interface

Group

TServer Properties

10.3.2.7.4.4  TServer.Profiles Property

Returns the profiles list.

```pascal
property Profiles: IProfiles;
```

See Also

IPhiles interface

Group

TServer Properties

10.3.2.7.4.5  TServer.License Property

Returns the current Server’s licence.

```pascal

```
property License: ILicense;

See Also
ILicense interface

Group
TServer Properties

10.3.2.7.4.6 TServer.Gateways Property

Returns the current Server's gateways.

Pascal
property Gateways: IGateways;

See Also
IGateway interface

Group
TServer Properties

10.3.2.8 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

10.3.2.9 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

### 10.4 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>
<tr>
<td>CRecorder</td>
<td>Session recording and playback.</td>
</tr>
<tr>
<td>CRecTracks</td>
<td>Collection of recorded session tracks.</td>
</tr>
<tr>
<td>CFileSystemFilter</td>
<td>Application FileSystem Virtualization.</td>
</tr>
<tr>
<td>CRegistryFilter</td>
<td>Application FileSystem Virtualization.</td>
</tr>
<tr>
<td>CHTMLDoc</td>
<td>Main class. Has methods, properties and events to allow to manage some web behavior.</td>
</tr>
</tbody>
</table>
10.4.1 CClientSettings Class

Allows to set some client settings.

Class Hierarchy

```
C++
class CClientSettings : public IClientSettings;
```

File

Thinfinity.VirtualUI.h

Group

C++ Classes

10.4.1.1 CClientSettings::CClientSettings Constructor

```
C++
CClientSettings(IVirtualUI * virtualUI);
```

Description

This is CClientSettings, a member of class CClientSettings.

Class

CClientSettings Class

10.4.1.2 CClientSettings::~CClientSettings Destructor

```
C++
~CClientSettings();
```

Description

This is ~CClientSettings, a member of class CClientSettings.

Class

CClientSettings Class
10.4.1.3 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 overload.</td>
</tr>
<tr>
<td>MouseMoveGestStyle</td>
<td>This is the overview for the MouseMoveGestStyle method overload.</td>
</tr>
</tbody>
</table>

10.4.1.3.1 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

10.4.1.3.1.1 CClientSettings::CursorVisible Method ()

Gets the the mouse pointer state.

C++

```cpp
bool CursorVisible();
```

Group
CursorVisible Method

10.4.1.3.1.2 CClientSettings::CursorVisible Method (bool)

Hides/shows the mouse pointer.

C++

```cpp
```
void CursorVisible(bool value);

Group

CursorVisible Method

10.4.1.3.2 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><code>CClientSettings::MouseMoveGestAction()</code></td>
<td>Gets the MouseMoveGestureAction value</td>
</tr>
<tr>
<td><code>CClientSettings::MouseMoveGestAction(MouseMoveGestureAction)</code></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>

Group

CClientSettings Methods

10.4.1.3.2.1 CClientSettings::MouseMoveGestAction Method ()

Gets the MouseMoveGestureAction value

C++

`MouseMoveGestureAction MouseMoveGestAction();`

Group

MouseMoveGestAction Method

10.4.1.3.2.2 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);`

Group

MouseMoveGestAction Method

10.4.1.3.3 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><code>CClientSettings::MouseMoveGestureStyle()</code></td>
<td>Gets the <code>MouseMoveGestureStyle</code> value</td>
</tr>
<tr>
<td><code>CClientSettings::MouseMoveGestureStyle(MouseMoveGestureStyle)</code></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**

10.4.1.3.3.1  `CClientSettings::MouseMoveGestureStyle Method ()`

Gets the `MouseMoveGestureStyle` value

**C++**

```cpp
MouseMoveGestureStyle MouseMoveGestureStyle();
```

### Group

**MouseMoveGestureStyle Method**

10.4.1.3.3.2  `CClientSettings::MouseMoveGestureStyle 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 MouseMoveGestureStyle(MouseMoveGestureStyle value);
```

### Group

**MouseMoveGestureStyle Method**

10.4.1.4  `CClientSettings Properties`

The properties of the `CClientSettings` class are listed here.

### Class

**CClientSettings Class**

### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>MouseMoveGestureStyle</code></td>
<td>This is <code>MouseMoveGestureStyle</code>, a member of class <code>CClientSettings</code>.</td>
</tr>
</tbody>
</table>
### CClientSettings::MouseMoveGestureStyle Property

```cpp
__property enum MouseMoveGestureStyle MouseMoveGestureStyle;
```

- **Description**
  This is `MouseMoveGestureStyle`, a member of class `CClientSettings`.

- **Group**
  `CClientSettings Properties`

### CClientSettings::MouseMoveGestureAction Property

```cpp
__property enum MouseMoveGestureAction MouseMoveGestureAction;
```

- **Description**
  This is `MouseMoveGestureAction`, a member of class `CClientSettings`.

- **Group**
  `CClientSettings Properties`

### CClientSettings::CursorVisible Property

```cpp
__property VARIANT_BOOL CursorVisible;
```

- **Description**
  This is `CursorVisible`, a member of class `CClientSettings`.

- **Group**
  `CClientSettings Properties`

### CBrowseInfo 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 a browser window on the end-user's application page.
Class Hierarchy

C++
class CBrowserInfo : public IBrowserInfo;

File
Thinfinity.VirtualUI.h

Group
C++ Classes

10.4.2.1 CBrowserInfo::CBrowserInfo Constructor

C++
CBrowserInfo(IVirtualUI * virtualUI);

Description
This is CBrowserInfo, a member of class CBrowserInfo.

Class
CBrowserInfo Class

10.4.2.2 CBrowserInfo::~CBrowserInfo Destructor

C++
~CBrowserInfo();

Description
This is ~CBrowserInfo, a member of class CBrowserInfo.

Class
CBrowserInfo Class

10.4.2.3 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>CustomData</td>
<td>This is the overview for the CustomData method overload.</td>
</tr>
<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>GetCookie</td>
<td>This is GetCookie, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>SetCookie</td>
<td>This is SetCookie, a member of class CBrowserInfo.</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>
<tr>
<td>SelectedRule</td>
<td>Returns the selected Browser Rule.</td>
</tr>
</tbody>
</table>

### 10.4.2.3.1 CustomData Method

This is the overview for the CustomData method overload.

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBrowserInfo::CustomData()</td>
<td>Returns custom application data</td>
</tr>
<tr>
<td>CBrowserInfo::CustomData(std::wstring)</td>
<td>Sets custom application data</td>
</tr>
</tbody>
</table>

## Group
CBrowserInfo Methods

10.4.2.3.1.1 CBrowserInfo::CustomData Method ()

Returns custom application data

C++

```cpp
std::wstring CustomData();
```

Group

CustomData Method

10.4.2.3.1.2 CBrowserInfo::CustomData Method (std::wstring)

Sets custom application data

C++

```cpp
void CustomData(std::wstring value);
```

Group

CustomData Method

10.4.2.3.2 ViewHeight Method

This is the overview for the ViewHeight method overload.

Overload List

<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

10.4.2.3.2.1 CBrowserInfo::ViewHeight Method ()

Returns the height of the VirtualUI Viewer.

C++

```cpp
int ViewHeight();
```
10.4.2.3.2  CBrowserInfo::ViewHeight Method (int)

Sets the height of the VirtualUI Viewer.

C++
void ViewHeight(int value);

Group
ViewHeight Method

10.4.2.3.3  ViewWidth Method

This is the overview for the ViewWidth method overload.

Group
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

10.4.2.3.3.1  CBrowserInfo::ViewWidth Method ()

Returns the width of the VirtualUI Viewer.

C++
int ViewWidth();

Group
ViewWidth Method

10.4.2.3.3.2  CBrowserInfo::ViewWidth Method (int)

Sets the width of the VirtualUI Viewer.

C++
void ViewWidth(int value);

Group
ViewWidth Method
10.4.2.3.4 CBrowserInfo::GetCookie Method

C++
__stdcall HRESULT GetCookie(BSTR Name, BSTR * Value);

Description
This is GetCookie, a member of class CBrowserInfo.

Group
CBrowserInfo Methods

10.4.2.3.5 CBrowserInfo::SetCookie Method

C++
__stdcall HRESULT SetCookie(BSTR Name, BSTR Value, BSTR Expires);

Description
This is SetCookie, a member of class CBrowserInfo.

Group
CBrowserInfo Methods

10.4.2.3.6 CBrowserInfo::BrowserWidth Method

C++
int BrowserWidth();

Description
Returns the width of the HTML element containing the VirtualUI Viewer.

Group
CBrowserInfo Methods

10.4.2.3.7 CBrowserInfo::BrowserHeight Method

Returns the height of the HTML element containing the VirtualUI Viewer.

C++
int BrowserHeight();

Group
CBrowserInfo Methods
10.4.2.3.8  CBrowserInfo::ScreenWidth Method

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

**C++**

```cpp
int ScreenWidth();
```

**Group**
CBrowserInfo Methods

10.4.2.3.9  CBrowserInfo::ScreenHeight Method

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

**C++**

```cpp
int ScreenHeight();
```

**Group**
CBrowserInfo Methods

10.4.2.3.10  CBrowserInfo::Username Method

Returns the logged-on Username.

**C++**

```cpp
std::wstring Username();
```

**Group**
CBrowserInfo Methods

10.4.2.3.11  CBrowserInfo::IPAddress Method

Returns the client's IP address.

**C++**

```cpp
std::wstring IPAddress();
```

**Group**
CBrowserInfo Methods

10.4.2.3.12  CBrowserInfo::UserAgent Method

Returns the browser's User Agent string.

**C++**

```cpp
std::wstring UserAgent();
```
10.4.2.3.13 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.

```cpp
std::wstring UniqueBrowserId();
```

10.4.2.3.14 CBrowserInfo::Location Method

Returns the URL of the current application.

```cpp
std::wstring Location();
```

10.4.2.3.15 CBrowserInfo::ScreenResolution Method

Returns the application screen resolution defined in the application profile.

```cpp
int ScreenResolution();
```

10.4.2.3.16 CBrowserInfo::Orientation Method

Returns the browser's orientation.

```cpp
Orientation Orientation();
```
10.4.2.3.17  CBrowserInfo::SelectedRule Method

Returns the selected Browser Rule.

```cpp
std::wstring SelectedRule();
```

Group

CBrowserInfo Methods

10.4.2.4  CBrowserInfo Properties

The properties of the CBrowserInfo class are listed here.

Class

CBrowserInfo Class

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewWidth</td>
<td>This is ViewWidth, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>ViewHeight</td>
<td>This is ViewHeight, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>BrowserWidth</td>
<td>This is BrowserWidth, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>BrowserHeight</td>
<td>This is BrowserHeight, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>ScreenWidth</td>
<td>This is ScreenWidth, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>ScreenHeight</td>
<td>This is ScreenHeight, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>Username</td>
<td>This is Username, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>IPAddress</td>
<td>This is IPAddress, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>UserAgent</td>
<td>This is UserAgent, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>ScreenResolution</td>
<td>This is ScreenResolution, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>Orientation</td>
<td>This is Orientation, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>UniqueBrowserId</td>
<td>This is UniqueBrowserId, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>CustomData</td>
<td>This is CustomData, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>Location</td>
<td>This is Location, a member of class CBrowserInfo.</td>
</tr>
<tr>
<td>SelectedRule</td>
<td>This is SelectedRule, a member of class CBrowserInfo.</td>
</tr>
</tbody>
</table>
10.4.2.4.1  CBrowserInfo::ViewWidth Property

```cpp
__property long ViewWidth;
```

**Description**

This is ViewWidth, a member of class CBrowserInfo.

**Group**

CBrowserInfo Properties

10.4.2.4.2  CBrowserInfo::ViewHeight Property

```cpp
__property long ViewHeight;
```

**Description**

This is ViewHeight, a member of class CBrowserInfo.

**Group**

CBrowserInfo Properties

10.4.2.4.3  CBrowserInfo::BrowserWidth Property

```cpp
__property HRESULT BrowserWidth;
```

**Description**

This is BrowserWidth, a member of class CBrowserInfo.

**Group**

CBrowserInfo Properties

10.4.2.4.4  CBrowserInfo::BrowserHeight Property

```cpp
__property HRESULT BrowserHeight;
```

**Description**

This is BrowserHeight, a member of class CBrowserInfo.

**Group**
CBrowserInfo Properties

10.4.2.4.5 CBrowserInfo::ScreenWidth Property

```cpp
__property HRESULT ScreenWidth;
```

**Description**

This is ScreenWidth, a member of class CBrowserInfo.

**Group**

CBrowserInfo Properties

10.4.2.4.6 CBrowserInfo::ScreenHeight Property

```cpp
__property HRESULT ScreenHeight;
```

**Description**

This is ScreenHeight, a member of class CBrowserInfo.

**Group**

CBrowserInfo Properties

10.4.2.4.7 CBrowserInfo::Username Property

```cpp
__property HRESULT Username;
```

**Description**

This is Username, a member of class CBrowserInfo.

**Group**

CBrowserInfo Properties

10.4.2.4.8 CBrowserInfo::IPAddress Property

```cpp
__property HRESULT IPAddress;
```

**Description**

This is IPAddress, a member of class CBrowserInfo.
10.4.2.4.9 CBrowserInfo::UserAgent Property

```cpp
__property HRESULT UserAgent;
```

**Description**

This is UserAgent, a member of class CBrowserInfo.

10.4.2.4.10 CBrowserInfo::ScreenResolution Property

```cpp
__property HRESULT ScreenResolution;
```

**Description**

This is ScreenResolution, a member of class CBrowserInfo.

10.4.2.4.11 CBrowserInfo::Orientation Property

```cpp
__property HRESULT Orientation;
```

**Description**

This is Orientation, a member of class CBrowserInfo.

10.4.2.4.12 CBrowserInfo::UniqueBrowserId Property

```cpp
__property HRESULT UniqueBrowserId;
```

**Description**
This is UniqueBrowserId, a member of class CBrowserInfo.

Group  
CBrowserInfo Properties

10.4.2.4.13  CBrowserInfo::CustomData Property

C++  
__property  BSTR CustomData;

Description  
This is CustomData, a member of class CBrowserInfo.

Group  
CBrowserInfo Properties

10.4.2.4.14  CBrowserInfo::Location Property

C++  
__property  HRESULT Location;

Description  
This is Location, a member of class CBrowserInfo.

Group  
CBrowserInfo Properties

10.4.2.4.15  CBrowserInfo::SelectedRule Property

C++  
__property  HRESULT SelectedRule;

Description  
This is SelectedRule, a member of class CBrowserInfo.

Group  
CBrowserInfo Properties

10.4.3  CDevServer Class

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

```cpp
class CDevServer : public IDevServer;
```

### File

Thinfinity.VirtualUI.h

### Group

**C++ Classes**

#### 10.4.3.1 CDevServer::CDevServer Constructor

```cpp
CDevServer(IVirtualUI * virtualUI);
```

### Description

This is CDevServer, a member of class CDevServer.

### Class

**CDevServer Class**

#### 10.4.3.2 CDevServer::~CDevServer Destructor

```cpp
~CDevServer();
```

### Description

This is ~CDevServer, a member of class CDevServer.

### Class

**CDevServer Class**

#### 10.4.3.3 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>StartBrowser</td>
<td>Instructs VirtualUI whether start or not the local web browser upon VirtualUI activation.</td>
</tr>
</tbody>
</table>

#### 10.4.3.3.1 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**

#### 10.4.3.3.1.1 CDevServer::Enabled Method ()

Gets the Development Server state.

```cpp
bool Enabled();
```

#### Group

**Enabled Method**

#### 10.4.3.3.1.2 CDevServer::Enabled Method (bool)

Enables/disables the Development Server.

```cpp
void Enabled(bool value);
```

#### Group

**Enabled Method**

#### 10.4.3.3.2 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><code>CDevServer::Port()</code></td>
<td>Gets the Development Server's TCP/IP listening port.</td>
</tr>
<tr>
<td><code>CDevServer::Port(int)</code></td>
<td>Sets the Development Server's TCP/IP listening port.</td>
</tr>
</tbody>
</table>

### Group

**CDevServer Methods**

10.4.3.3.2.1  CDevServer::Port Method ()

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

```cpp
int Port();
```

### Group

**Port Method**

10.4.3.3.2.2  CDevServer::Port Method (int)

Sets the Development Server's TCP/IP listening port.

```cpp
void Port(int value);
```

### Group

**Port Method**

10.4.3.3.3  CDevServer::StartBrowser Method

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

```cpp
void StartBrowser(bool value);
```

### Group

**CDevServer Methods**

10.4.3.4  CDevServer Properties

The properties of the CDevServer class are listed here.

### Class
## CDevServer Class

### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>This is Enabled, a member of class CDevServer.</td>
</tr>
<tr>
<td>Port</td>
<td>This is Port, a member of class CDevServer.</td>
</tr>
<tr>
<td>StartBrowser</td>
<td>This is StartBrowser, a member of class CDevServer.</td>
</tr>
</tbody>
</table>

#### 10.4.3.4.1 CDevServer::Enabled Property

```cpp
__property VARIANT_BOOL Enabled;
```

**Description**
This is Enabled, a member of class CDevServer.

**Group**
CDevServer Properties

#### 10.4.3.4.2 CDevServer::Port Property

```cpp
__property long Port;
```

**Description**
This is Port, a member of class CDevServer.

**Group**
CDevServer Properties

#### 10.4.3.4.3 CDevServer::StartBrowser Property

```cpp
__property VARIANT_BOOL StartBrowser;
```

**Description**
This is StartBrowser, a member of class CDevServer.

**Group**
CDevServer Properties
10.4.4  JSObject Class

Represents a custom remotable object.

Class Hierarchy

C++

```c++
class JSObject : public 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 vice versa, 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

10.4.4.1  JSObject::JSObject Constructor

```c++
JSObject(std::wstring id);
```

Description

This is JSObject, a member of class JSObject.

Class

JSObject Class

10.4.4.2  JSObject::~JSObject Destructor

```c++
~JSObject();
```

Description
This is ~JSObject, a member of class JSObject.

### Class

**JSObject Class**

#### 10.4.4.3 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><code>FireEvent</code></td>
<td>This is the overview for the FireEvent method overload.</td>
</tr>
<tr>
<td><code>Id</code></td>
<td>This is the overview for the Id method overload.</td>
</tr>
<tr>
<td><code>QueryInterface</code></td>
<td>This is QueryInterface, a member of class JSObject.</td>
</tr>
<tr>
<td><code>ApplyChanges</code></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><code>ApplyModel</code></td>
<td>Propagates the whole JSObject definition to the javascript client.</td>
</tr>
<tr>
<td><code>Properties</code></td>
<td>List containing all properties of this object.</td>
</tr>
<tr>
<td><code>Methods</code></td>
<td>List containing all methods of this object.</td>
</tr>
<tr>
<td><code>Events</code></td>
<td>List containing all events of this object.</td>
</tr>
<tr>
<td><code>Objects</code></td>
<td>List containing all events of this object.</td>
</tr>
</tbody>
</table>

#### 10.4.4.3.1 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><code>JSObject::FireEvent (BSTR, IJSArguments *)</code></td>
<td>This is FireEvent, a member of class JSObject.</td>
</tr>
<tr>
<td><code>JSObject::FireEvent (std::wstring, IJSArguments*)</code></td>
<td>This is FireEvent, a member of class JSObject.</td>
</tr>
</tbody>
</table>
### JSObject Methods

#### 10.4.4.3.1.1 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

#### 10.4.4.3.1.2 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

#### 10.4.4.3.2 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

#### 10.4.4.3.2.1 JSObject::Id Method ()

Identifier of the object. It must be unique among siblings objects.

**C++**
```cpp
std::wstring Id();
```
void Id(std::wstring value);

## Group

### Id Method

10.4.4.3.3 JSObject::QueryInterface Method

```cpp
__stdcall HRESULT QueryInterface(REFIID rid, void ** ppvObject);
```

### Description

This is QueryInterface, a member of class JSObject.

## Group

### JSObject Methods

10.4.4.3.4 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.

```cpp
__stdcall HRESULT ApplyChanges();
```

## Group

### JSObject Methods

10.4.4.3.5 JSObject::ApplyModel Method

Propagates the whole JSObject definition to the javascript client.

```cpp
__stdcall HRESULT ApplyModel();
```

## Group

### JSObject Methods

10.4.4.3.6 JSObject::Properties Method

List containing all properties of this object.

```cpp
CJSProperties* Properties();
```

## Group
JSObject Methods

10.4.4.3.7  JSObject::Methods Method

List containing all methods of this object.

```cpp
CJSMethods* Methods();
```

Group

JSObject Methods

10.4.4.3.8  JSObject::Events Method

List containing all events of this object.

```cpp
CJSEvents* Events();
```

Group

JSObject Methods

10.4.4.3.9  JSObject::Objects Method

List containing all events of this object.

```cpp
CJSObjects* Objects();
```

Group

JSObject Methods

10.4.4.4  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>This is Id, a member of class JSObject.</td>
</tr>
<tr>
<td>![Properties]</td>
<td>This is Properties, a member of class JSObject.</td>
</tr>
</tbody>
</table>
10.4.4.1  JSObject::Id Property

C++
__property BSTR Id;

- **Description**
  This is Id, a member of class JSObject.

- **Group**
  JSObject Properties

10.4.4.2  JSObject::Properties Property

C++
__property HRESULT Properties;

- **Description**
  This is Properties, a member of class JSObject.

- **Group**
  JSObject Properties

10.4.4.3  JSObject::Methods Property

C++
__property HRESULT Methods;

- **Description**
  This is Methods, a member of class JSObject.

- **Group**
  JSObject Properties

10.4.4.4  JSObject::Events Property

C++
__property HRESULT Events;
Description

This is Events, a member of class JSObject.

Group

JSObject Properties

10.4.4.4.5 JSObject::Objects Property

```cpp
__property HRESULT Objects;
```

Description

This is Objects, a member of class JSObject.

Group

JSObject Properties

10.4.5 JSCallback Class

Class Hierarchy

```
JSObject
  JSCallback
```

```cpp
class JSCallback : public IJSCallback;
```

File

Thinfinity.VirtualUI.h

Description

This is class JSCallback.

Group

C++ Classes

10.4.5.1 JSCallback::JSCallback Constructor

```cpp
JSCallback(JSMethodeCallback Proc);
```

Description
This is JSCallback, a member of class JSCallback.

Class

JSCallback Class

10.4.5.2 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>

10.4.5.2.1 JSCallback::Callback Method

C++

```
__stdcall HRESULT Callback(IJSObject* Parent, IJSMethod* Method);
```

Description

This is Callback, a member of class JSCallback.

Group

JSCallback Methods

10.4.6 JSBinding Class

Class Hierarchy

```
  IJSBinding
   |   JSBinding
```

C++

```
class JSBinding : public IJSBinding;
```

File

Thinfinity.VirtualUI.h

Description

This is class JSBinding.
10.4.6.1 JSBinding::JSBinding Constructor

```cpp
JSBinding(JSPropertySet Proc);
```

**Description**
This is JSBinding, a member of class JSBinding.

**Class**
JSBinding Class

10.4.6.2 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>

10.4.6.2.1 JSBinding::Set Method

```cpp
__stdcall HRESULT Set(IJSObject* Parent, IJSPROPERTY* Prop);
```

**Description**
This is Set, a member of class JSBinding.

**Group**
JSBinding Methods

10.4.7 VirtualUI

**Group**
C++ Classes

10.4.8 CRecorder Class

Session recording and playback.

Class Hierarchy

```
     Recorder
        └── CRecorder
```

C++
class CRecorder : public IRecorder;

File
Thinfinity.VirtualUI.h

Group
C++ Classes

10.4.8.1 CRecorder::CRecorder Constructor

```
C++
CRecorder(IVirtualUI * virtualUI);
```

Description
This is CRecorder, a member of class CRecorder.

Class
CRecorder Class

10.4.8.2 CRecorder::~CRecorder Destructor

```
C++
~CRecorder();
```

Description
This is ~CRecorder, a member of class CRecorder.

Class
CRecorder Class
10.4.8.3 CRecorder Methods

The methods of the CRecorder class are listed here.

Class

CRecorder Class

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>This is the overview for the Filename method overload.</td>
</tr>
<tr>
<td>Options</td>
<td>This is the overview for the Options method overload.</td>
</tr>
<tr>
<td>Position</td>
<td>Returns the current entry of the session currently playing.</td>
</tr>
<tr>
<td>Count</td>
<td>Returns the number of entries in the loaded session file for playback.</td>
</tr>
<tr>
<td>State</td>
<td>Reports the recorder state:</td>
</tr>
<tr>
<td></td>
<td>- Inactive: Recorder is idle.</td>
</tr>
<tr>
<td></td>
<td>- Recording: Record in progress.</td>
</tr>
<tr>
<td></td>
<td>- Playing: Playing a session.</td>
</tr>
<tr>
<td>Tracks</td>
<td>Returns the list of tracks in a session file (for playback purposes only).</td>
</tr>
<tr>
<td>Rec</td>
<td>Begins session recording.</td>
</tr>
<tr>
<td>Play</td>
<td>Begins session playback.</td>
</tr>
<tr>
<td>Stop</td>
<td>Stops the current recording.</td>
</tr>
</tbody>
</table>

10.4.8.3.1 Filename Method

This is the overview for the Filename method overload.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRecorder::Filename ()</td>
<td>Returns the current filename.</td>
</tr>
<tr>
<td>CRecorder::Filename (std::wstring)</td>
<td>Sets the filename for session recording. Is the path and name of the session file to be used. No extension is needed; the recorder will generate two files, with extensions idx and dat.</td>
</tr>
</tbody>
</table>

Group

CRecorder Methods
10.4.8.3.1.1 CRecorder::Filename Method ()

Returns the current filename.

**C++**

```cpp
std::wstring Filename();
```

**Group**

Filename Method

10.4.8.3.1.2 CRecorder::Filename Method (std::wstring)

Sets the filename for session recording. Is the path and name of the session file to be used. No extension is needed; the recorder will generate two files, with extensions idx and dat.

**C++**

```cpp
void Filename(std::wstring value);
```

**Group**

Filename Method

10.4.8.3.2 Options Method

This is the overview for the Options method overload.

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRecorder::Options ()</td>
<td>This is Options, a member of class CRecorder.</td>
</tr>
<tr>
<td>CRecorder::Options (long)</td>
<td>This is Options, a member of class CRecorder.</td>
</tr>
</tbody>
</table>

**Group**

CRecorder Methods

10.4.8.3.2.1 CRecorder::Options Method ()

**C++**

```cpp
long Options();
```

**Description**

This is Options, a member of class CRecorder.

**Group**

Options Method
10.4.8.3.2.2 CRecorder::Options Method (long)

```cpp
void Options(long value);
```

**Description**

This is Options, a member of class CRecorder.

**Group**

Options Method

10.4.8.3.3 CRecorder::Position Method

Returns the current entry of the session currently playing.

```cpp
long Position();
```

**Group**

CRecorder Methods

10.4.8.3.4 CRecorder::Count Method

Returns the number of entries in the loaded session file for playback.

```cpp
long Count();
```

**Group**

CRecorder Methods

10.4.8.3.5 CRecorder::State Method

Reports the recorder state:
- Inactive: Recorder is idle.
- Recording: Record in progress.
- Playing: Playing a session.

```cpp
RecorderState State();
```

**Group**

CRecorder Methods
10.4.8.3.6 CRecorder::Tracks Method

Returns the list of tracks in a session file (for playback purposes only).

```cpp
CRecTracks* Tracks();
```

Group

CRecorder Methods

10.4.8.3.7 CRecorder::Rec Method

Begins session recording.

```cpp
void Rec(std::wstring Label);
```

Description

Label: track name. This name will allow to play different recordings in the same session.

Group

CRecorder Methods

10.4.8.3.8 CRecorder::Play Method

Begins session playback.

```cpp
__stdcall HRESULT Play(long From, long To);
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>long From</td>
<td>Entry of playback start.</td>
</tr>
<tr>
<td>long To</td>
<td>Last entry to play.</td>
</tr>
</tbody>
</table>

Remarks

To play a recorded session:

- Set the Filename property with the path and name of the session file to be played (the file with idx extension).
- Call Play with the range of entries to play.

To play an entire session, pass 0 and the Count property.

To play only a specific track, pass the Position of track to reproduce as From, and the Position of next track as To. For the last track, the To parameter must be the Count property of recorder.
10.4.8.3.9 CRecorder::Stop Method

Stops the current recording.

```cpp
__stdcall HRESULT Stop();
```

10.4.8.4 CRecorder Properties

The properties of the CRecorder class are listed here.

### Class

**CRecorder Class**

#### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracks</td>
<td>This is Tracks, a member of class CRecorder.</td>
</tr>
</tbody>
</table>

10.4.8.4.1 CRecorder::Tracks Property

```cpp
__property HRESULT Tracks;
```

### Description

This is Tracks, a member of class CRecorder.

10.4.9 CRecTracks Class

Collection of recorded session tracks.
```cpp
class CRecTracks : public IRecTracks;
```

**File**

Thinfinity.VirtualUI.h

**Group**

**C++ Classes**

### 10.4.9.1 CRecTracks::CRecTracks Constructor

```cpp
CRecTracks(IRecorder* Recorder);
```

**Description**

This is CRecTracks, a member of class CRecTracks.

**Class**

**CRecTracks Class**

### 10.4.9.2 CRecTracks::~CRecTracks Destructor

```cpp
~CRecTracks () ;
```

**Description**

This is ~CRecTracks, a member of class CRecTracks.

**Class**

**CRecTracks Class**

### 10.4.9.3 CRecTracks Methods

The methods of the CRecTracks class are listed here.

**Class**

**CRecTracks Class**

**Public Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Returns a session track.</td>
</tr>
</tbody>
</table>
10.4.9.3.1 CRecTracks::Item Method

Returns a session track.

```cpp
    #include "RecTracks.h"

    #ifdef _USE_PUB_SUB
        #include "RecTracksPubSub.h"
    #endif

    #ifdef _USE_PERSIST
        #include "RecTracksPersist.h"
    #endif

    #ifdef _USE_DB
        #include "RecTracksDatabase.h"
    #endif

    #ifdef _USE_FILE
        #include "RecTracksFile.h"
    #endif

    class CRecTracks : public CBaseTracks
    {
    public:
        #ifdef _USE_PUB_SUB
            CActiveObject *pubSubObj;
        #else
            CActiveObject pubSubObj;
        #endif

        #ifdef _USE_PERSIST
            CActiveObject *persistentObj;
        #else
            CActiveObject persistentObj;
        #endif

        #ifdef _USE_DB
            CActiveObject *databaseObj;
        #else
            CActiveObject databaseObj;
        #endif

        #ifdef _USE_FILE
            CActiveObject *fileObj;
        #else
            CActiveObject fileObj;
        #endif

        // Constructor
        CRecTracks();

        // Destructor
        ~CRecTracks();

        // Count Method
        long Count();

        // Item Method
        IRecTrack *Item(long index);
    
    private:
        // Member variables
        // ...
    
    #ifdef _USE_PUB_SUB
        #include "RecTracksPubSubImpl.h"
    #endif

    #ifdef _USE_PERSIST
        #include "RecTracksPersistImpl.h"
    #endif

    #ifdef _USE_DB
        #include "RecTracksDatabaseImpl.h"
    #endif

    #ifdef _USE_FILE
        #include "RecTracksFileImpl.h"
    #endif

    }; // CRecTracks
```

10.4.9.3.2 CRecTracks::Count Method

Number of tracks

```cpp
    long Count();
```

10.4.9.4 CRecTracks Properties

The properties of the CRecTracks class are listed here.

```cpp
    class CRecTracks : public CBaseTracks
    {
    public:
        #ifdef _USE_PUB_SUB
            CActiveObject *pubSubObj;
        #else
            CActiveObject pubSubObj;
        #endif

        #ifdef _USE_PERSIST
            CActiveObject *persistentObj;
        #else
            CActiveObject persistentObj;
        #endif

        #ifdef _USE_DB
            CActiveObject *databaseObj;
        #else
            CActiveObject databaseObj;
        #endif

        #ifdef _USE_FILE
            CActiveObject *fileObj;
        #else
            CActiveObject fileObj;
        #endif

        // Constructor
        CRecTracks();

        // Destructor
        ~CRecTracks();

        // Count Method
        long Count();

        // Item Method
        IRecTrack *Item(long index);
    
    private:
        // Member variables
        // ...
    
    #ifdef _USE_PUB_SUB
        #include "RecTracksPubSubImpl.h"
    #endif

    #ifdef _USE_PERSIST
        #include "RecTracksPersistImpl.h"
    #endif

    #ifdef _USE_DB
        #include "RecTracksDatabaseImpl.h"
    #endif

    #ifdef _USE_FILE
        #include "RecTracksFileImpl.h"
    #endif

    }; // CRecTracks
```

10.4.9.4.1 CRecTracks::Item Property

```cpp
    #pragma pack(push, 8)
    __property HRESULT Item;
    #pragma pack(pop)
```

**Description**

This is Item, a member of class CRecTracks.

**Group**

CRecTracks Properties
10.4.10 CFileSystemFilter Class

Application FileSystem Virtualization.

### Class Hierarchy

```
| IFileSystemFilter | CFileSystemFilter |
```

C++

```cpp
class CFileSystemFilter : private IFileSystemFilter;
```

### File

`Thinfinity.VirtualUI.h`

### Group

C++ Classes

### 10.4.10. CFileSystemFilter::CFileSystemFilter Constructor

```cpp
CFileSystemFilter(IVirtualUI * virtualUI);
```

### Description

This is `CFileSystemFilter`, a member of class `CFileSystemFilter`.

### Class

`CFileSystemFilter Class`

### 10.4.10. CFileSystemFilter::~CFileSystemFilter Destructor

```cpp
~CFileSystemFilter();
```

### Description

This is `~CFileSystemFilter`, a member of class `CFileSystemFilter`.

### Class

`CFileSystemFilter Class`
10.4.10. CFileSystemFilter Methods

The methods of the CFileSystemFilter class are listed here.

Class

CFileSystemFilter Class

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>This is the overview for the Active method overload.</td>
</tr>
<tr>
<td>CfgFile</td>
<td>This is the overview for the CfgFile method overload.</td>
</tr>
<tr>
<td>User</td>
<td>This is the overview for the User method overload.</td>
</tr>
</tbody>
</table>

10.4.10.3.1 Active Method

This is the overview for the Active method overload.

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFileSystemFilter::Active()</td>
<td>Returns the filesystem virtualization's state.</td>
</tr>
<tr>
<td>CFileSystemFilter::Active(bool)</td>
<td>Sets the filesystem virtualization state for the container application.</td>
</tr>
</tbody>
</table>

Group

CFileSystemFilter Methods

10.4.10.3.1.1 CFileSystemFilter::Active Method ()

Returns the filesystem virtualization's state.

```
C++
bool Active();
```

Group

Active Method

10.4.10.3.1.2 CFileSystemFilter::Active Method (bool)

Sets the filesystem virtualization state for the container application.

```
C++
void Active(bool value);
```
10.4.10.3.2 CfgFile Method

This is the overview for the CfgFile method overload.

<table>
<thead>
<tr>
<th>Overload List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>CFileSystemFilter::CfgFile ()</td>
</tr>
<tr>
<td>CFileSystemFilter::CfgFile (std::wstring)</td>
</tr>
</tbody>
</table>

10.4.10.3.2.1 CFileSystemFilter::CfgFile Method ()

Returns the configuration filename.

```cpp
std::wstring CfgFile();
```

10.4.10.3.2.2 CFileSystemFilter::CfgFile Method (std::wstring)

Sets the configuration filename for the filesystem virtualization.

```cpp
void CfgFile(std::wstring value);
```

10.4.10.3.3 User Method

This is the overview for the User method overload.

<table>
<thead>
<tr>
<th>Overload List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>CFileSystemFilter::User ()</td>
</tr>
</tbody>
</table>
CFileSystemFilter::User (std::wstring)
Sets the user for the filesystem virtualization.

#### Group

**CFileSystemFilter Methods**

10.4.10.3.3.1 CFileSystemFilter::User Method ()

Returns the current user.

```cpp
std::wstring User();
```

#### Group

**User Method**

10.4.10.3.3.2 CFileSystemFilter::User Method (std::wstring)

Sets the user for the filesystem virtualization.

```cpp
void User(std::wstring value);
```

#### Group

**User Method**

10.4.10. CFileSystemFilter Properties

The properties of the CFileSystemFilter class are listed here.

#### Class

**CFileSystemFilter Class**

#### Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>This is User, a member of class CFileSystemFilter.</td>
</tr>
<tr>
<td>CfgFile</td>
<td>This is CfgFile, a member of class CFileSystemFilter.</td>
</tr>
<tr>
<td>Active</td>
<td>This is Active, a member of class CFileSystemFilter.</td>
</tr>
</tbody>
</table>
10.4.10.4.1 CFileSystemFilter::User Property

```cpp
__property BSTR User;
```

Description
This is User, a member of class CFileSystemFilter.

Group
CFileSystemFilter Properties

10.4.10.4.2 CFileSystemFilter::CfgFile Property

```cpp
__property BSTR CfgFile;
```

Description
This is CfgFile, a member of class CFileSystemFilter.

Group
CFileSystemFilter Properties

10.4.10.4.3 CFileSystemFilter::Active Property

```cpp
__property VARIANT_BOOL Active;
```

Description
This is Active, a member of class CFileSystemFilter.

Group
CFileSystemFilter Properties

10.4.11 CRegistryFilter Class

Application FileSystem Virtualization.

Class Hierarchy
```
| RegistryFilter | CRegistryFilter |
```

```cpp
class CRegistryFilter : private IRegistryFilter;
```
File
Thinfinity.VirtualUI.h

Group
C++ Classes

10.4.11. CRegistryFilter::CRegistryFilter Constructor

C++
CRegistryFilter(IVirtualUI * virtualUI);

Description
This is CRegistryFilter, a member of class CRegistryFilter.

Class
CRegistryFilter Class

10.4.11. CRegistryFilter::~CRegistryFilter Destructor

C++
~CRegistryFilter();

Description
This is ~CRegistryFilter, a member of class CRegistryFilter.

Class
CRegistryFilter Class

10.4.11. CRegistryFilter Methods

The methods of the CRegistryFilter class are listed here.

Class
CRegistryFilter Class

Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

© 2018, Cybele Software, Inc.
10.4.11.3.1 Active Method

This is the overview for the Active method overload.

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CRegistryFilter::Active()</code></td>
<td>Returns the registry virtualization's state.</td>
</tr>
<tr>
<td><code>CRegistryFilter::Active(bool)</code></td>
<td>Sets the registry virtualization state for the container application.</td>
</tr>
</tbody>
</table>

### Group

CRegistryFilter Methods

10.4.11.3.1.1 CRegistryFilter::Active Method ()

Returns the registry virtualization's state.

```
C++
bool Active();
```

### Group

Active Method

10.4.11.3.1.2 CRegistryFilter::Active Method (bool)

Sets the registry virtualization state for the container application.

```
C++
void Active(bool value);
```

### Group

Active Method

10.4.11.3.2 CfgFile Method

This is the overview for the CfgFile method overload.

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CRegistryFilter::CfgFile Method()

Returns the configuration filename.

```cpp
std::wstring CfgFile();
```

CRegistryFilter::CfgFile Method (std::wstring)

Sets the configuration filename for the registry virtualization.

```cpp
void CfgFile(std::wstring value);
```

User Method

This is the overview for the User method overload.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRegistryFilter::User</td>
<td>Returns the current user.</td>
</tr>
<tr>
<td>CRegistryFilter::User</td>
<td>Sets the user for the registry virtualization.</td>
</tr>
</tbody>
</table>

CRegistryFilter Methods
std::wstring User();

Group

User Method

10.4.11.3.3.2 CRegistryFilter::User Method (std::wstring)

Sets the user for the registry virtualization.

C++

void User(std::wstring value);

Group

User Method

10.4.11. CRegistryFilter Properties

The properties of the CRegistryFilter class are listed here.

Class

CRegistryFilter Class

Public Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>This is User, a member of class CRegistryFilter.</td>
</tr>
<tr>
<td>CfgFile</td>
<td>This is CfgFile, a member of class CRegistryFilter.</td>
</tr>
<tr>
<td>Active</td>
<td>This is Active, a member of class CRegistryFilter.</td>
</tr>
</tbody>
</table>

10.4.11.4.1 CRegistryFilter::User Property

C++

__property BSTR User;

Description

This is User, a member of class CRegistryFilter.

Group

CRegistryFilter Properties
10.4.11.4.2 CRegistryFilter::CfgFile Property

```c++
__property BSTR CfgFile;
```

**Description**
This is CfgFile, a member of class CRegistryFilter.

**Group**
CRegistryFilter Properties

10.4.11.4.3 CRegistryFilter::Active Property

```c++
__property VARIANT_BOOL Active;
```

**Description**
This is Active, a member of class CRegistryFilter.

**Group**
CRegistryFilter Properties

10.4.12 CHTMLDoc Class

Main class. Has methods, properties and events to allow to manage some web behavior.

**Class Hierarchy**

```c++
class CHTMLDoc : public IHTMLDoc;
```

**File**
Thinfinity.VirtualUI.h

**Group**
C++ Classes

10.4.12. CHTMLDoc::CTMMLDoc Constructor

```c++
CTMMLDoc(IVirtualUI* virtualUI);
```
### Description
This is CHTMLDoc, a member of class CHTMLDoc.

### Class
CHTMLDoc Class

#### 10.4.12. CHTMLDoc::~CHTMLDoc Destructor

```
C++
~CHTMLDoc () ;
```

### Description
This is ~CHTMLDoc, a member of class CHTMLDoc.

### Class
CHTMLDoc Class

#### 10.4.12. CHTMLDoc Methods

The methods of the CHTMLDoc class are listed here.

### Class
CHTMLDoc Class

#### Public Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateSessionURL</td>
<td>Creates an url pointing to a local filename. This url is valid during the session lifetime and its private to this session.</td>
</tr>
<tr>
<td>CreateComponent</td>
<td>Inserts an HTML. Used to insert regular HTML elements or WebComponents with custom elements.</td>
</tr>
<tr>
<td>GetSafeURL</td>
<td>Returns a safe, temporary and unique URL to access any local file.</td>
</tr>
<tr>
<td>LoadScript</td>
<td>Loads a script from URL. If Filename is specified, creates a session URL first and then load the script from that Filename.</td>
</tr>
<tr>
<td>ImportHTML</td>
<td>Imports an HTML from URL. If Filename is specified, creates a session URL first and then imports the html file from that Filename.</td>
</tr>
</tbody>
</table>
10.4.12.3.1 CHTMLDoc::CreateSessionURL Method

Creates an url pointing to a local filename. This url is valid during the session lifetime and its private to this session.

**C++**
```cpp
void CreateSessionURL(std::wstring Url, std::wstring Filename);
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>Arbitrary relative url.</td>
</tr>
<tr>
<td>filename</td>
<td>Local filename</td>
</tr>
</tbody>
</table>

**Group**

CHTMLDoc Methods

10.4.12.3.2 CHTMLDoc::CreateComponent Method

Inserts an HTML. Used to insert regular HTML elements or WebComponents with custom elements.

**C++**
```cpp
void CreateComponent(std::wstring Id, std::wstring Html, INT64 ReplaceWnd);
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>std::wstring Id</td>
<td>ID to assign to the main element of the HTML to be inserted</td>
</tr>
<tr>
<td>std::wstring Html</td>
<td>HTML snippet</td>
</tr>
<tr>
<td>INT64 ReplaceWnd</td>
<td>Wnd to be replaced and tied to</td>
</tr>
</tbody>
</table>

**Remarks**

When ReplaceWnd is <> 0 and points to a valid window handle, the positioning of the main element will follow the Wnd positioning, simulating an embedding.

**Group**

CHTMLDoc Methods

10.4.12.3.3 CHTMLDoc::GetSafeURL Method

Returns a safe, temporary and unique URL to access any local file.

**C++**
```cpp
std::wstring GetSafeURL(std::wstring Filename, int Minutes);
```
Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>std::wstring Filename</td>
<td>Local filename</td>
</tr>
<tr>
<td>int Minutes</td>
<td>Expiration in minutes</td>
</tr>
</tbody>
</table>

Group

CHTMLDoc Methods

10.4.12.3.4 CHTMLDoc::LoadScript Method

Loads a script from URL. If Filename is specified, creates a session URL first and then load the script from that Filename.

C++

```cpp
void LoadScript(std::wstring Url, std::wstring Filename);
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>std::wstring Url</td>
<td>Relative URL</td>
</tr>
<tr>
<td>std::wstring Filename</td>
<td>Local filename (optional)</td>
</tr>
</tbody>
</table>

Group

CHTMLDoc Methods

10.4.12.3.5 CHTMLDoc::ImportHTML Method

Imports an HTML from URL. If Filename is specified, creates a session URL first and then imports the html file from that Filename.

C++

```cpp
void ImportHTML(std::wstring Url, std::wstring Filename);
```

Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>std::wstring Url</td>
<td>Relative URL</td>
</tr>
<tr>
<td>std::wstring Filename</td>
<td>Local filename (optional)</td>
</tr>
</tbody>
</table>
10.5 VirtualUI.sdk.min.js

<table>
<thead>
<tr>
<th>Files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>thinfinity.virtualui.doc.js</td>
<td>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.</td>
</tr>
<tr>
<td>thinfinity.jsro.doc.js</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>

10.5.1 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
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>

#### 10.5.1.1 Events

#### 10.5.1.1.1 onLoading

**JavaScript**

```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

#### 10.5.1.2 onShow

**JavaScript**

```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

**File**

*Thinfinity.VirtualUI*

#### 10.5.1.3 onClose

**JavaScript**

```javascript
virtualUI.onClose = function(){
    //...TODO
};
```

### Description

Occurs when 'close' is triggered. The event will be fired once the app is closed.

### File

**File**

*Thinfinity.VirtualUI*

#### 10.5.1.4 onError

**JavaScript**

```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

**File**

*Thinfinity.VirtualUI*
10.5.1.2 Properties

10.5.1.2.1 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

### 10.5.2 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**

```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><code>start</code></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><code>on</code></td>
<td>Allows you to add an event to a specific node.</td>
</tr>
<tr>
<td><code>off</code></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><code>model</code></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>

10.5.2.1 Methods

10.5.2.1.1 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*

10.5.2.1.2 on

```javascript
// The events can be:
// Local (client side)
JsRO.on('model:Form1.checkBox1','changed',function(elem){
    //yourCode
});
// Remote (are the 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**

#### 10.5.2.1.3 off

**JavaScript**

```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 added and FALSE when it was not.

### Description

 Allows you to remove an event from a specific node.

### File

**Thinfinity.JsRO**

© 2018, Cybele Software, Inc.
10.5.2.2 Properties

10.5.2.2.1 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`