

Context App Tool



Addon Development Manual

Author: Michael Jordon

Email: cat@contextis.com

Web: <http://cat.contextis.com>



Contents

| | |
|---------------------------------|-----------|
| 1 Introduction | 3 |
| 2 Basic AddOns Interface | 4 |
| 3 User Controls | 5 |
| 3.1 AppTool.HTTPEdit | 5 |
| 3.2 AppTool.HTTPLog | 6 |
| 3.2.1 Custom Columns | 7 |
| 3.3 AppTool.HTTPView | 8 |
| 3.4 AppTool.Browser.WebBrowser | 8 |
| 3.5 AppTool.RichEditor | 9 |
| 4 Core Classes | 11 |
| 4.1 HTTPType | 11 |
| 4.2 SendRequest | 15 |
| 5 Examples | 17 |
| 5.1 User Controls | 17 |
| 5.2 Sample Plugin | 18 |



1 Introduction

Context App Tool (CAT) is an application to facilitate manual web application penetration testing. CAT allows the development of additional AddOn modules to further extend the CAT functionality. This document details the API that is accessible to the Addons. This API allows for the use of the UI controls that are used across CAT and also the underlying infrastructure for build and sending HTTP requests.

CAT is a freely available tool and the concept of the AddOns is to allow the core of CAT to be augmented with the addition of extra modules which are added as new tab types on the main CAT interface.

CAT has been developed over the years in response to job-by-job requirements and therefore has grown organically. This means that there are aspects of the API which could be improved and better represented. Improvements have been made during the development of the Addons interface, which will hopefully make development of CAT Addons less complex.



2 Basic AddOns Interface

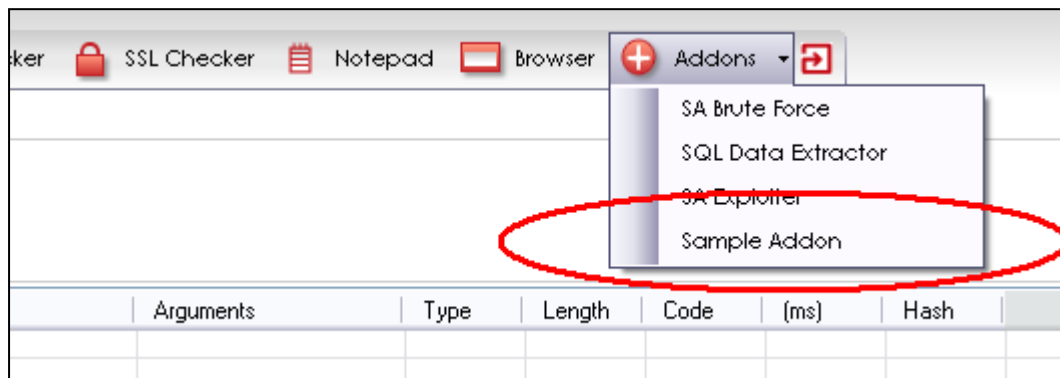
CAT enumerates the 'addons' directory under the installed CAT directory and checks all DLLs to see if they contain any classes that inherit from the AppTool.MainPanel class. If there are classes which match, then the GetPanelName() method is called to get the name of the panel. A menu item is then added to the Add-ons dropdown list on the main toolbar in CAT. A user can then select the addon which will add a new panel to CAT which displays the addon control..

The first step is to add a reference to the apptool.exe binary within Visual Studio. For example the following code will create a blank addon:

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

namespace SampleAddon
{
    public class SamplePanel : AppTool.MainPanel
    {
        public override String GetPanelName()
        {
            return "Sample Addon";
        }
    }
}
```

When this DLL is compiled and then placed in the addons directory CAT will detect it on loading and add a button to the addons tool bar as per the next screenshot:





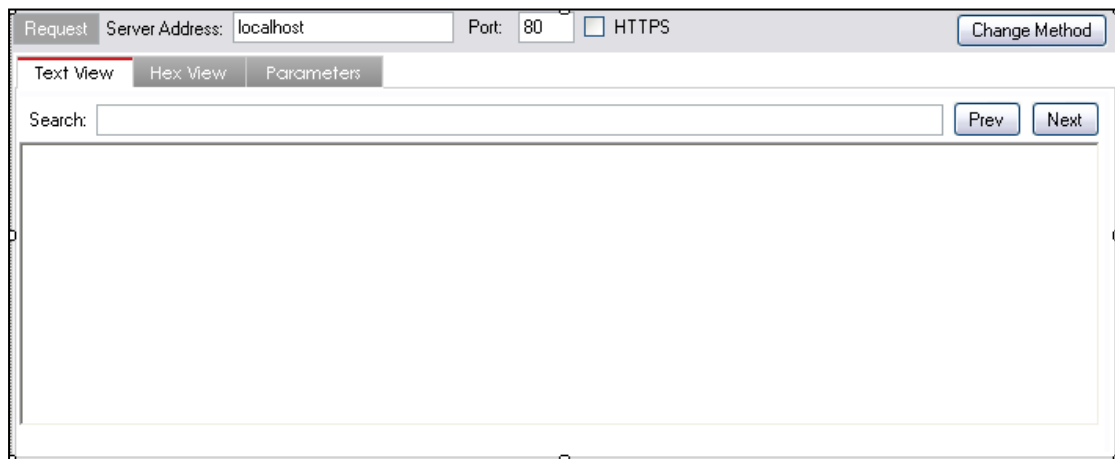
3 User Controls

The following user controls are the main controls that are exposed to addons and are the core controls used within CAT to interact with the user:

- HTTPEdit – Allows editing of a HTTP request
- HTTPLog – Lists HTTP responses
- HTTPView – Displays all the information about a HTTP request and response
- WebBrowser – Integrated web browser.
- RichTextEditor – Text editor with integrated conversion options.

3.1 AppTool.HTTPEdit

This control allows for http requests to be edited by a user.



The HTTPEdit control exposes the following public methods:

- `AutoUpdateContentLength` - A Boolean field that determines whether the the HTTP Content Length will automatically be updated when requesting the user's data.
- `AutoStripHttpHeaders` - A Boolean field that controls whether the compression support headers are removed or not.
- `ChangedEventHandler(object sender, EventArgs e)` - An even that is fired when the data in the edit is changed.
- `ShowParametersTab` - Enable/disable the parameters tab.
- `ShowHexTab` - Enable/disable the hex view tab
- `static String UpdateContentLengthString(String data)` - A generic http length updating method.
- `static String UpdateContentLengthString(String data,bool AddOnZero)`: An overload for the above where if the `AddOnZero` is true the function will add a content-length header even when there is no content length. This is used if the method requires a content length like a POST or PUT.



- `public void Stop()` – Halts any threads that are performing actions on the HTTPLog e.g. auto tests, spider.
- `public void AddHTTPData(HTTPType h)` - Adds a HTTPType to the log.
- `public void AddHTTPData(HTTPType h, CustomColumnValue c)` – Adds a HTTPType and also a custom column for the entry (see custom columns).
- `public void AddHTTPData(HTTPType h, CustomColumnValue[] cols)` – Adds a HTTPType to the log and multiple custom columns.
- `public void AddHTTPData(HTTPType h, CustomColumnValue[] cols, bool usetag)` – Adds an HTTPType with custom columns and will use the tag identifier in the HTTPType to overwrite any log item if it is already in the log list.
- `public void ClearLog()` – Clears the log.
- `public void SetAutoFollow(bool f)` – Sets whether the log should scroll to the bottom when a new log item is added.
- `public HTTPTypeStore GetHTTPRequests()` – Gets the HTTPTypeStore object which contains all the requests in the log.
- `public void AddColumn(String colname, int width)` – Creates a new column see ('Custom Columns').
- `public void RemoveColumn(String name)` – Removes a custom column and not a main column.
- `public void AddHTTPDataStore(HTTPTypeStore addstore)` – Populates the list object with the HTTPTypeStore object.
- `public void HideSameSiteColumns()` – Where an HTTPLog is being used for requests on only the same site, this call will simplify the view to remove the columns which will not change.
- `public XmlNode GetXML()` – Retrieves a XML representation of the HTTPLog. This does not include the actual log entries, just the state of the user control and a name of the log file which contains the log file entries.
- `Public void SetXML(XmlNode xmlNode)` – Uses the XML provided to configure the settings with the XML provided. Also loads the log items from the file specified in the XML.
- `public void LoadLogFile(string name)` – Populates the log file with the contents of the file specified.

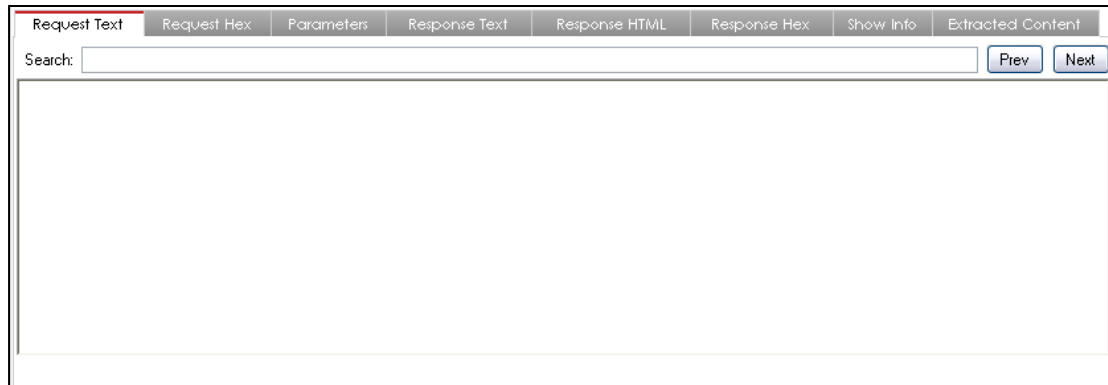
3.2.1 Custom Columns

To add extra columns to a HTTPLog the calling code must create a CustomColumnValue object which has three properties of name, value and back colour. The name of the column will identify into which column the value is to be added and must have been created with AddColumn. The 'value' is the contents of the cell for that row and the back colour will colour the background for that specific cell. See SQL and XSS auto testers within CAT for examples of where this is used. A row can have multiple custom columns in which case they are passed as an array in any order.



3.3 AppTool.HTTPView

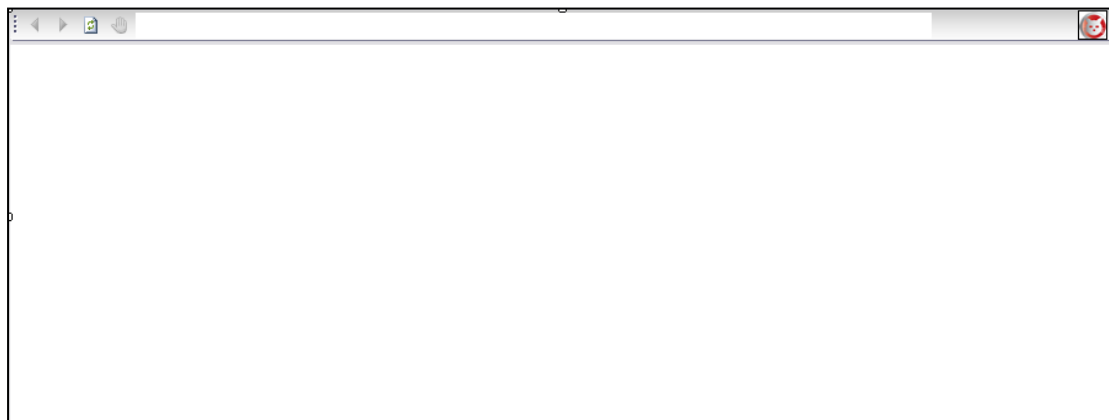
The HTTPView object is used to display a request and/or response for an HTTPType object. The control will populate the text, hex, HTML Show Info etc. It can also be used to edit an HTTPType object.



- `public bool ReadOnly` – Determines whether a user can edit the request and response.
- `public void SetHTTPData(HTTPType h)` – Sets the HTTPType value.
- `public void Clear()` – Clears the contents of all the tabs.
- `public static String FormatHTML(String s)` – A static function for converting text HTML into RTF format.
- `public HTTPType GetHTTPData()` – Retrieves the HTTPType object from the control. This includes any changes made by the user if it is not ReadOnly.
- `virtual public void AddTab(TabPage newpage)` – Adds a new tab to the view to augment the results.
- `public void SelectTab(String name)` – Selects a TAB page to be shown to the user.

3.4 AppTool.Browser.WebBrowser

The integrated WebBrowser is an IE control that renders HTML and can be fully interactive. It can also be used to set the web browser's proxy.

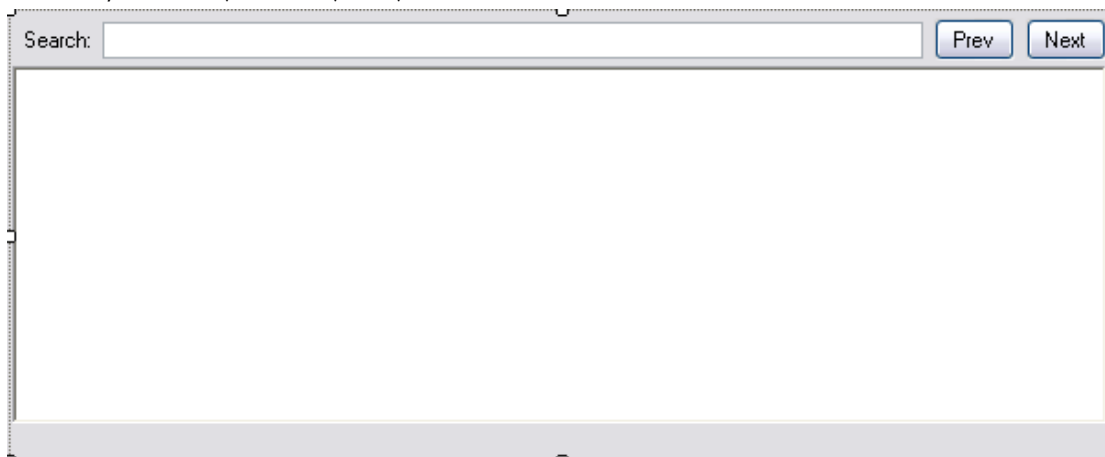




- public bool Silent – Prevents certain pop-ups from appearing
- public bool Loaded – Returns true when the page has loaded.
- public void Navigate(String URL) – Loads the URL in the web browser.
- public void Navigate(Uri u) – Loads the URI in the web browser.
- public void SetHTML(String html,String host,int port,bool SSL) – Sets the web browser to render the HTML. The host, port and SSL are set also so that supporting components (JS, PNG, GIF etc.) can also be loaded.
- public void SetHTML(String html) – Renders the HTML with no supporting components.
- public void Clear() – Clears the control.
- public void NoProxy() – Sets the browser to use no proxy.
- public void SetProxy(String URL) – Sets the web browser to use the specified proxy. The URL should be in the form: http://localhost:8085 to use the localhost and port 8085.
- public bool ShowControls – Shows or hides the top bar with the navigation controls.
- public HtmlDocument GetDom() – This is used to retrieve a HtmlDocument object from the control. This allows CAT to use Internet Explorer to convert HTML into a object. For example the Spider will call 'SetHTML()' while the control is not visible, and then wait until the Loaded property is true before calling GetDom() to retrieve a structured form of that HTML. This HtmlDocument can then easily be queried to extract out information about the HTML, e.g. links, forms. Invalid HTML will be handled as per Internet Explorer.

3.5 AppTool.RichEditor

The RichTextEditor is used by a user to enter/edit text with the help of a right click context menu for performing conversations that can aid in web application related text. This is the standard menu which includes conversations for example URL encode/decode, base64, Hex, etc





- `public const String CLIPBOARD_HTTP_SINGLE = "APPTOOL-HTTP-SINGLE"` – This constant is used to specify the clipboard format for transferring requesting a single HTTPType object. The object used the GetXML representation.
- `public const String CLIPBOARD_HTTP_MULTI = "APPTOOL-HTTP-MULTI"` – The constant for multiple HTTPType object in the clipboard.
- `public delegate void ChangedEventHandler(object sender, EventArgs e)` - The handler for notifications that the user has changed the contents of the control.
- `public event ChangedEventHandler Changed` – The event for the change of the contents of the control.
- `public delegate bool PasteEventHandler(object sender, EventArgs e)` – Handler class for being notified that the user has pasted into the control.
- `public event PasteEventHandler PasteEvent` – The event for notification that the user has pasted into the control.
- `public void SetText(String s)` – Sets the plain text of the control.
- `public void AppendText(String s)` – Adds text to the end of the control.
- `public String GetText()` – Retrieves the plain text of the control.
- `public void EnableMarkRegion()` – Enables the user to mark regions in the request that must be post-encoded by the Repeater. For example, it allows the user to set the background colour to 'base64' - the request will return that section encoded.
- `public void SetRtf(String r)` – Set the Rich Text Format for the control. Allows for colours, images, size etc. It uses the standard RTF format codes.
- `public String GetRtf()` – Retrieve the RTF formatted content.
- `public void Search()` – Causes a search to be performed on the key word entered by the user.
- `public bool Empty()` – Returns a Boolean of whether the control contains any text.
- `public void Clear()` – Delete all the contents of the control.
- `public static String BytesToHex(byte[] bs)` – Converts a byte array into a hex string representation e.g. AA = 4141
- `public static String BytesToUnicodeHex(byte[] bs)` – Converts a byte array into a Unicode hex representation e.g. AA = 00410041 (currently only supports 8 bit input values).



4 Core Classes

The code for processing the request and response of HTTP data are contained in the following classes:

- `AppTool.HttpConnections.HTTPType` – Contains the request and response data.
- `AppTool.HttpConnections.SendRequest` – Used to send HTTPTypes and receive a response.
- `AppTool.HttpConnections.Cookie` – Cookie object containing the name, value and host for a HTTP cookie.
- `AppTool.HttpConnections.HTTPParameter` – Name/value and method of a HTTP parameter. Methods can be GET (for URL based parameters), POST (for URL name value pairs in the body) and MIME (multi-part MIME body).
- `AppTool.DiffForm` – Visual diff between two strings shown in a form. The constructor takes two strings and can then be used to show a window containing the visual difference between the two.

4.1 HTTPType

HTTPType is the core class for processing data, and it contains both HTTP requests and responses. The GUI controls all use HTTPType to get and set requests and responses. HTTPEdit returns this type, HTTPView displays it, and HTTPLog shows a list of them. HTTPType provides a range of methods to extract various data from the request and also to modify the various components of a request.

- `public int tag` – A identifier that is used to track HTTPType requests. It is used in the HTTPLog to allow items to be updated (see HTTPLog for more details). The user can specify their integer for this value.
- `public bool boolStripHttpHeaders = true` – Removes certain HTTP headers i.e. compression Accept values.
- `public HTTPType()` – Blank constructor.
- `public HTTPType(HTTPType h)` – Creates a clone of another HTTPType.
- `public HTTPType(XmlNode x)` – Create a HTTPType based on the XML data.
- `public void SetSSL(Boolean s)` – Specifies whether the request uses SSL.
- `public void SetHost(String h)` – Sets the IP address or domain whether the request should be sent. This does not change the HTTP Host header.
- `public String GetMethod()` – Gets the HTTP method used e.g. GET, POST.
- `public void SetMethod(String s)` – Sets the HTTP method to use.
- `public int GetResponseLength()` – Get the length of the response including headers.
- `public int GetPort()` – Gets the TCP port to be used/was used for the server.
- `public String GetHost()` – Gets the IP address or domain of the server (not from the HTTP header).



- `public String GetURL()` – Gets the complete URL e.g. `http://domain:23/a.php?a=b`
- `public String GetBaseURL()` – Gets the URL without the path or parameters e.g. `http://domain:23/`
- `public String GetArgsLine()` – Gets the arguments line for both GET and POST requests e.g. `a=b`.
- `public String GetPath()` – Gets the path of the file e.g. `/a.php`.
- `public void SetPath(String p)` – Sets the path.
- `public string GetProtocol()` – Gets either HTTP or HTTPS depending on whether SSL is being used.
- `public byte[] GetResponse()` – Gets the raw bytes of the response including headers.
- `public Boolean GetSSL()` – Returns true if SSL is set.
- `public String GetResponseStr()` – Gets a string representation of the response.
- `public byte[] GetRequest()` – Gets the raw bytes of the request.
- `public String GetRequestStr()` – Gets a string representation of the request.
- `public String GetRequestHeaderStr()` – Gets a string containing the request headers.
- `public byte[] GetRequestBody()` – Gets the raw bytes of the body of the request.
- `public DateTime GetWhenRequestSet()` – A DateTime of when the request was sent.
- `public DateTime GetWhenResponseSet()` – A DateTime of When the response was sent.
- `public void SetWhenRequestSet(DateTime d)` – Sets the request sent time.
- `public String GetStatusCode()` – Gets the response code value e.g. 200, 302, 500.
- `public int GetDurationMs()` – The time taken to receive a response in milliseconds.
- `public string GetGetParametersAsString()` – Gets the parameter on the GET line e.g. `a=b`;
- `public string GetPostParametersAsString()` – Gets the name value pair POST parameters e.g. `a=b`.
- `public String GetRequestCookieLine()` – Returns the cookie line sent to the server e.g. `Cookie: phpsession=xyz`
- `public HTTPType Copy()` – Returns a copy of the HTTPType object.
- `public bool IsRedirect()` – Returns true if the response was a redirect e.g. 301, 302.
- `public void SetRequest(String inRequest)` – Sets the request text.



- `public void SetRequest(HTTPType inRequest)` – Sets the Request from another HTTPType.
- `public void SetRequest(byte[] inRequest)` – Sets the request from a binary array.
- `public void SetRequest(byte[] inRequest,int len)` – Sets the request from a binary array but only up to a certain length.
- `public void SetResponse(byte[] inResponse)` – Sets the response from a binary array.
- `public void SetResponse(byte[] inResponse, int len)` – Sets the response from a binary array only up to a certain length.
- `public void SetResponse(String s)` – Sets the response from a string.
- `public void SetPort(int inport)` – Sets the TCP port to be used when sending the request.
- `public String GetResponseMIMEType()` – Returns the content encoding type of the response e.g. text/html.
- `public string GetExtension()` – Returns the file extension e.g. php.
- `public byte[] GetResponseBody()` – Gets the body of the response in a binary format.
- `public void SetMessage(String s)` – Sets a status message from the object, this can include the current status of the object and errors during sending.
- `public String GetMessage()` – Returns the current status message of the object.
- `public void SetError(bool e)` – Sets whether the HTTPType has an error, used when a exception occurred during sending the request or receiving the response.
- `public bool GetError()` – Returns true if the HTTPType object is set in error mode. `GetMessage()` includes the details of the failure.
- `public List<Cookie> GetCookies()` – Returns a list of Cookie objects from the request (see cookies for more details).
- `public CookieCollection GetCookieCollection()` – Returns a collection of cookie objects (see cookies for more details).
- `public void SetCookies(CookieCollection c)` – Sets the Cookies for the request (see cookies for more details).
- `public Boolean IsSimpleType()` – Returns true if the file extension is of gif, jpg, png, ico, css, js, axd, vsxd and the request has not parameters.
- `public String GetViewState()` – Returns the viewstate from the response.
- `public String GetComments()` – Returns a string with the comments from the response.
- `public XmlNode GetXML()` – Gets a XML representation of the HTTPType.
- `public void SetXML(XmlNode n)` – Sets the properties of the HTTPType based on the XmlNode.



- `public void SetXML(XmlNode n, bool DeepRead)` – Sets the HTTPType properties of the XmlNode. If DeepRead is true then any embedded HTTPTypes are also recreated. Embedded nodes are from AutoTests which can contain many more HTTPTypes. For performance it is often not required to perform this level of creation.
- `public bool SetXML(String s)` – Creates a HTTPType from a String containing a XML representation of the HTTPType.
- `public bool SetXML(string s, bool DeepRead)` – Creates an HTTPType based on the string of XML. DeepRead will also recreate any embedded HTTPTypes which can be from auto tests.
- `public void SetUri(Uri u)` – Uses a .NET Uri object to set the request text. The request will be a GET to the URL in the Uri object.
- `public void SetUriBuildRequest(Uri u, CookieCollection cookies)` – Creates a HTTP request to the URL in the Uri and also sets the cookies as per the collection.
- `public void SetGetLine(string p)` – Sets the GET arguments for the request.
- `public string[] GetRequestHeaders()` – Returns a list of all the request headers in a string array.
- `public String GetBodyHash()` – Returns a MD5 hash of the body of the response. This does not include the HTTP headers.
- `public String GetResponseBodyStr()` – Gets the body of the response as a string.
- `public String GetResponseHeaderStr()` – Gets the headers of the response as a string.
- `public String [] GetParametersEchoed()` – Returns any parameters URL/BODY/MIME that are contained in the body of the response.
- `public List<HTTPParameter> GetParameters()` – Returns all the parameters in the request (URL/BODY/MIME) as an HTTPParameter list.
- `public void SetParameters(List<HTTPParameter> NewParameters)` – Sets the HTTP request with the parameters in the list.
- `public HTTPParameter GetParameter(String Name)` – Gets the parameter of the name specified or returns NULL if not found.
- `public void SetParameter(HTTPParameter NewParameter)` – Sets a single parameter to the value specified. If the parameter does not exist then it is created.
- `public List<String> GetLinks()` – returns a list of all the URLs in either anchor tags or forms. This is performed using regex and it is recommended that the web browser is used for a more accurate extraction of links.
- `public List<Uri> GetUriLinks()` – Returns the all links as per GetLinks but as a Uri object.
- `public Uri GetUri()` – returns the Uri .NET object of the HTTP request.



- `public static String[] GetRequestHeaders(String input)` – Returns the HTTP headers in the request as strings.
- `public void SetHTTPHeader(String name,String value)` – Sets a specific HTTP request header with the value specified. The name can be with or without the colon. If the name does not exist in the headers then the request is not changed.
- `public string GetResponseHTTPHeaderValue(string p)` – Returns the value of the HTTP response header or NULL if not found.
- `public string GetRequestHTTPHeaderValue(string p)` – Returns the value of the HTTP request value or NULL if not found.
- `public void ClearAutoTestResults()` – Removes all auto tests that are associated with the HTTPType.
- `public bool HasResponse()` – Returns true if the HTTPType contains a response.

4.2 SendRequest

SendRequest object performs the actual sending of a HTTP request and handling of the response. The following methods are exposed:

- `public SendRequest(HTTPType intype, int timeoutMs)` – The main constructor to create a SendRequest object for sending the HTTPType object's request to the server as specified within the HTTPType object. The timeout is the length in time in milliseconds that the request will wait for a response from the server.
- `public SendRequest(String domain, int Port, String data, bool ssl, int timeoutMS)` – Create a SendRequest object based on the domain, port, SSL and timeout (in milliseconds) values specified. The data is the string of the request to be sent.
- `public SendRequest(String domain, int Port, byte[] data, bool ssl, int timeoutMS)` – Creates a SendRequest object based on the domain, port, ssl and timeout (in milliseconds) values specified. The data is the raw bytes of the HTTP request.
- `public SendRequest(String URL)` – Uses the URL to determine the target server and creates a GET request to that address. Will throw an exception if the URL is not correctly formatted.
- `public void Send(AsyncCallback GotData,AsyncCallback GotError)` – This is the main method for actually sending the HTTP Request to the server and receiving the response. The call is Asynchronise and will call back to either GotData if the call completes successfully or GetError if there was a communications failure. See examples below on how to send request.
- `public HTTPType GetHTTPData()` – Returns the current HTTPType object associated with the SendRequest object. If the request has been sent then this will include the response.
- `public void SetTag(int i)` – Sets the tag for the HTTPType that is being used for the SendRequest. See HTTPType for more details about tags and tracking requests.



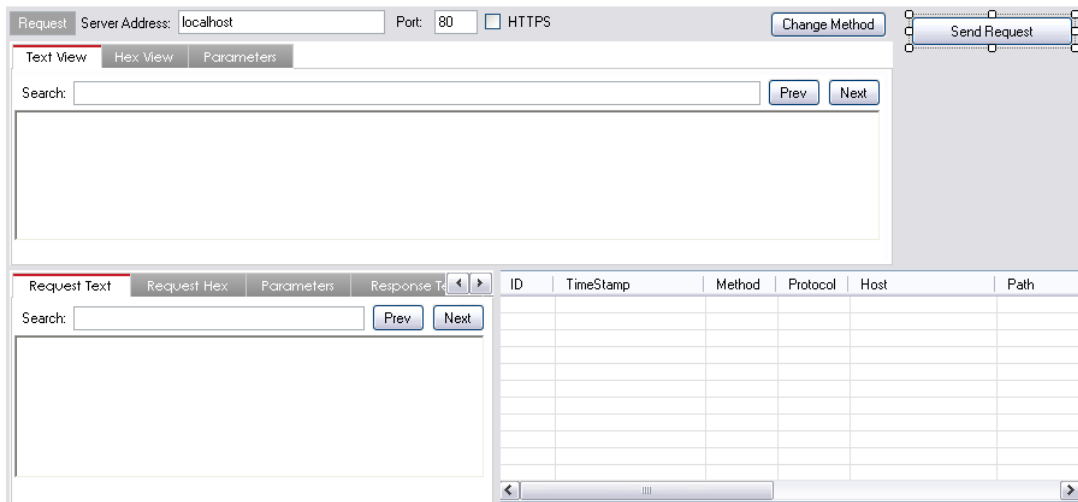
- `public int GetTag()` – Return the current tag.
- `public void FollowRedirect(HTTPType OriginalH, AsyncCallback inGotData, AsyncCallback inGotError)` – Request the `SendRequest` object to retrieve the result of the redirect e.g. a request that results in a 302 HTTP status code. This request will request the Location URL specified.
- `public void SetSendOptions(AppTool.HttpConnections.SendRequestsOptions o)` - `SendRequestsOptions` is a object which contains global settings that are normally configured under the file->options menu. This includes upstream proxies, NTLM authentication etc. These settings can be changed using this option.
- `public HTTPType SendWait()` – This will perform the same as send request but will be Synchronise. Therefore the method will not return until either the server has responded to the request or an error as occurred. This could take up to the timeout specified or the duration of a response.



5 Examples

5.1 User Controls

This example explains how to create a basic repeater using the controls provided. First step is to add the AppTool components to the toolbox by using Choose Items from the dropdown menu and selecting the AppTool.exe. Then add an HTTPEdit, HTTPView, HTTPLog and a button to the user control that inherits from AppTool.MainPanel as seen above. As can be seen in the following screenshot.



The code that is required in the onclick event for the 'Send Request' button, to send the request and then add the result to both the HTTPView and the HTTPLog is the following:

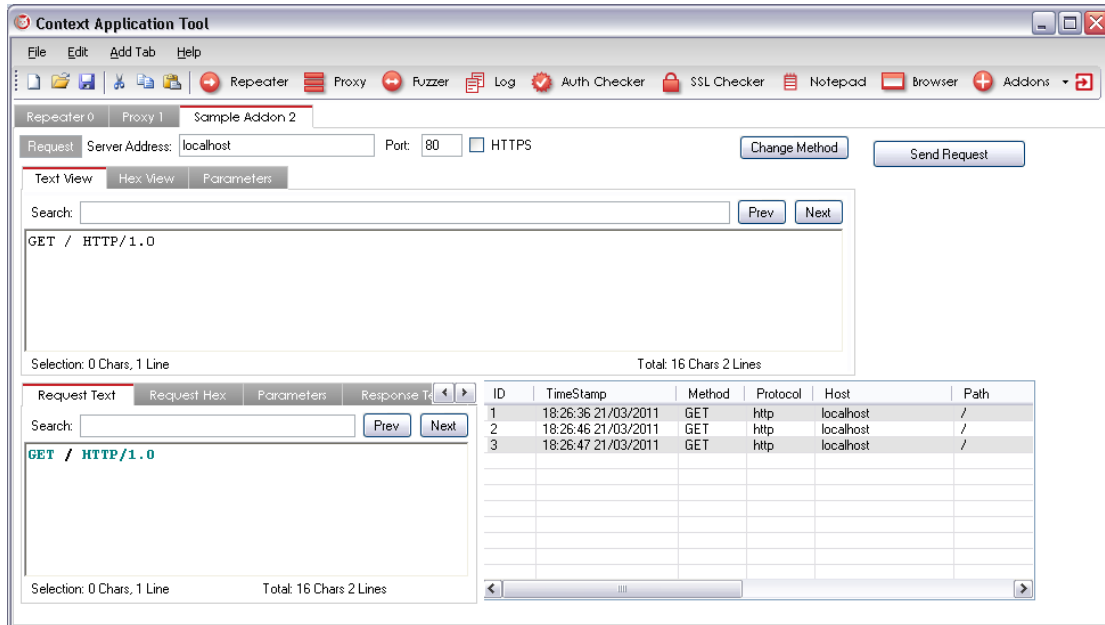
```
private void ButtonSendRequest_Click(object sender, EventArgs e)
{
    //Get the request from the user control
    HTTPType Request = httpEdit1.GetHTTPData();

    //create a send request object with a 30sec timeout
    SendRequest sr = new SendRequest(Request, 30 * 1000);

    //Send the request synchronously, in the real world
    //use asynchronously or the UI thread will block
    HTTPType RequestResponse = sr.SendWait();

    //Now add the response to the user controls
    httpView1.SetHTTPData(RequestResponse);
    httpLog1.AddHTTPData(RequestResponse);
}
```

At this point a basic repeat has been created. By compiling this code into a DLL and placing the DLL into the AddOns directory as detailed above the following screenshot shows the code inaction:



5.2 Sample Plugin

The sample plugin provides further examples of how to interact with the various aspects of CAT. To compile the sample you must add a reference to the AppTool.exe from the installation directory.