'Network Connectivity Solutions'	**Necoso** Het Kasteel 315 7325 PE Apeldoorn The Netherlands	**Phone:** +31-(0)55-3601410 **Fax:** +31-(0)84-7246122 **Website:** www.necoso.com **E-mail:** info@necoso.com

ID1021

AN 001 - Adapting web content for ID1021

Project	ID1021	**Document ID**	Application Note
Customer	<Customer>	**Version**	2.0
Classification	Public	**Status**	Final
Author(s)	Robert Hulsebos	**Date**	01-jan-2004

Summary

This document contains instructions and help for engineers that are about to create or customize web content for the ID1021 internet communications coprocessor from Necoso.

Document History

Version	Date	Author(s)	Description
1.0	04-apr-2002	Robert Hulsebos	First draft version
1.1	13-jan-2003	Robert Hulsebos	Updated for ID1021 firmware v1.6. Added paragraph about internationalizing web pages.
2.0	01-jan-2004	Robert Hulsebos	Updated for ID1021 firmware v2.1. Added paragraph about internationalizing web pages. Necoso update.

Document Distribution

Version	Date	To	Company
1.0	04-apr-2002	-major customers-	
1.1	13-jan-2003	-same as previous version-	
2.0	01-jan-2004	-same as previous version-	

Table of Contents

Table of tables

Table of figures

1 Introduction

This document is an application note for engineers who want to create/adapt the web content (HTML pages, picture files, etc) that are to be used with the HTTP server of the ID1021 internet communication coprocessor from Necoso.

Normally, the ID1021 is integrated into an OEM device for realizing a web based user interface for the device. The ID1021 enables an end-user of the OEM device to use a web browser (e.g. Microsoft Internet Explorer or Netscape Navigator) for monitoring/controlling the OEM device over an ethernet network. For this the ID1021 firmware includes an embedded HTTP server, FTP server and file system driver. Together they allow for the HTML pages of the OEM device's user interface to be created and tested on a PC or workstation elsewhere. When this process is finished, the resulting HTML files can be downloaded to the flash disk of the ID1021. The integrated HTTP server of the ID1021 will then present these HTML files when called upon by a HTTP client. (web browser)

Although there are no HTML format-specific limitations as to the contents of the HTML files for the ID1021, there are certain ID1021-specific aspects that must be kept in mind when creating/adapting HTML files for the ID1021. This is important, because the applications running inside the ID1021 may depend on these aspects. Ignoring these aspects may result in unexpected results or corrupted user interfaces. This document describes the HTML programming aspects.

1.1 Scope

This document focuses on the HTML programming aspects of the ID1021 only.

1.2 Intended audience

This document was intended for web design/software engineers that are responsible creating/adapting the HTML pages for the ID1021. Basic knowledge of the ID1021, the HTTP protocol and the HTML language are assumed.

1.3 Feed back

This application note document is a 'living' document for the benefit of all ID1021 users and developers. It will grow with the passing of time. It is based on our and your hands-on experiences while developing web based user interface solutions with the ID1021. Please send any suggestions for improvements, hints and comments to the author – they will be appreciated.

1.4 Terms and abbreviations

The table below contains an alphabetical list of the terms and abbreviations used in this document.

Term/abbreviation	Description
ASCII	American Standard Code for Information Interchange, defines character-set symbols for characters with value in range 0 – 127.
ASP	Active Server Page, Microsoft definition for dynamic HTML page, i.e. a HTML page for which content may vary in time.
EFS	Embedded File System
ESA	Embedded Server Application. Application format for ID1021 applications which are permanently active. See also ISA.
Firmware	In-product software, in the context of this document: the software that is stored within the boundaries of the ID1021 housing.
FAQ	Frequently Asked questions

FTP	File Transfer Protocol, as specified by RFC 959.
HTML	Hyper Text Markup Language. Format for HTML pages (also called 'web pages'), as specified by RFC 1866.
HTTP	Hyper Text Transfer Protocol. Communication protocol, as specified by RFC 1945, and used by for example a web browser for retrieving a web page from a web server. (HTTP server)
IP	Internet Protocol, as specified by RFC 791
ISA	Internet Server Application. Application format for ID1021 applications that are only active when called upon by the ID1021 HTTP server. See also ESA.
ID1021	Internet communications coprocessor from Necoso.
OEM	Original Equipment Manufacturer
PC	Personal Computer
RFC	Request For Comment. Normally RFC <number> refers to an internet protocol specification document with number <number>. The RFC documents are publicly available and can be downloaded from website *http://www.ietf.org/rfc/* RFC 1880 is an overview document and contains an overview of all available Internet Standards and their most up to data RFCs.
TCP	Transmission Control Protocol, as specified by RFC 793.
Telnet	Telnet protocol, as specified by RFC 854.
URL	Uniform Resource Locator, specification for location of a web page. Contains internet address of webserver and file name of web page on that webserver. E.g. *192.168.0.1/index.htm*
Web content	Collection of HTML files, picture files and other files that may be request by an HTTP client (web browser) when connected to a HTTP server.

Table 1: Terms and abbreviations

1.5 References

The following table lists the documents that are referenced in this document.

Reference	**Document ID**	**Version**	**Description/title**
[INSTMAN]	Installation manual.doc	2.0	*ID1021 installation manual for OEM customers.*
[ID1021]	ID1021.PDF	2.0	*ID1021 datasheet, can be downloaded from www.necoso.com*
[HTTP]	RFC1945.TXT	May 1996	*"Hypertext Transfer Protocol -- HTTP/1.0"*
[HTML]	RFC1866.TXT	V2.0	*"Hypertext Markup Language v2.0"*

Table 2: Referenced documents

1.6 Overview

This document is organized in chapters and (sub)paragraphs. Chapters are used for dividing aspects into major aspect groups. Paragraphs and subparagraphs are used to elaborate on details aspects within a specific aspect group.

Chapter 1 introduces this document.
Chapter 2 describes the basic architecture of the ID1021 integrated HTTP server.
Chapter 3 describes what to do and not to do when customizing HTML for the ID1021. Also contains some hints for testing and extending the HTML user interface.

2 Architecture of ID1021 embedded HTTP server

2.1 Architecture description

Before we can elaborate on the aspect of customizing HTML for the ID1021 we must first understand the basic architecture of the ID1021 integrated HTTP server.

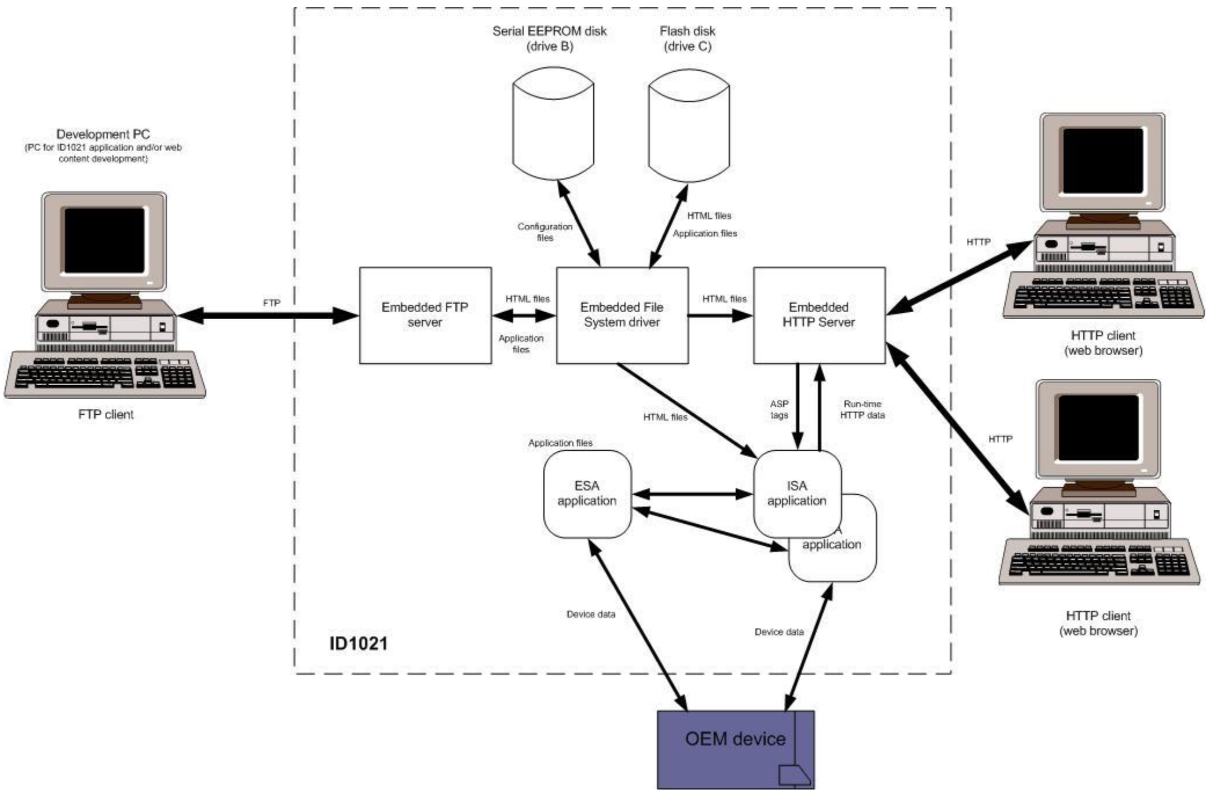

Figure 1: Architecture of ID1021 embedded HTTP server

The figure above outlines the basic ID1021 architecture. A short description of each of the items depicted in this figure:

Development PC : This represents the PC or workstation that is used to do the application and/or web content development for the ID1021. The resulting application and web content files are transferred to the ID1021 using the FTP protocol. The development PC runs the FTP client, the ID1021 the FTP server. On the ID1021 the FTP server communicates with the Embedded File System (EFS) driver for storing/retrieving files to/from the flash disk and serial EEPROM disk.

Flash disk: This is the ID1021 internal solid state disk that is based on flash EEPROM technology. Due to its block-erase characteristics it is less suitable for storage of volatile data, i.e. data that changes a lot. Therefore with the ID1021 it is normally used for storing non-volatile data such as application files and web content. (HTML pages, picture files, etc)
With the standard ID1021 the serial EEPROM disk is about 2 MB in size.

Serial EEPROM disk: This is the ID1021 internal solid state disk that is base on serial EEPROM technology. Due to its byte-erase characteristics it is suitable for storage of volatile data, i.e. that is often reprogrammed. (e.g. configuration data) Due to its relatively slow serial access characteristics it is less suitable for storage of application files and web content.

With the standard ID1021 the serial EEPROM disk is about 8 KB in size, so relatively small.

Embedded FTP server: This driver implements the server part of the FTP protocol. It communicates with a FTP client on the development PC for transfer of file data. For storage/retrieval of file data it communicates with the Embedded File System driver.

Embedded File System (EFS) driver: This is the driver that implements a file system for the flash disk and the serial EEPROM disk so that the disk can be accessed using open/close/read/write operations at file level.

ISA applications: ISA stands for Internet Server Application. ISA applications are executable programs that are only active ('executed') when called upon by the HTTP server. As opposed to ESA applications, which are permanently active after power on of the ID1021.The HTTP server will only call an ISA application when it is requested by an HTTP client. (usually a web browser)
ISA applications are normally used to gather run-time data and present the data as part of a (dynamic) HTML page. For example an HTML page that contains the OEM device temperature value, an ISA may be called to get the current temperature from the OEM device and insert it in the HTTP data stream to the HTTP client. The temperature value may be retrieved directly from the OEM device itself or may be derived from an ESA application. (that has integrated OEM device driver for communications with the OEM device)
Another difference between ISA and ESA applications is the number of instantiations that may occur. Of one ISA application multiple instantiations may be active at the same time. (when multiple HTTP clients are requesting the same web page calling upon the ISA application) All instantiations share the same code, but have their own set of application data. Of one ESA application only one instance is active at any moment in time.
ISA applications are stored on the flash disk of the ID1021.
A special ISA application is the default.isa application. This ISA application is called by the HTTP server for client requests of files with the .htm filename extension. The default.isa application implements access security for HTML pages and interprets the password.txt file if present on the flash disk, see [INSTMAN] for more details.

ESA applications: ESA stands for Embedded Server Application. An ESA application is an application that is permanently active, i.e. it is started when the ID1021 is powered on and is never stopped. Normally, ESA applications contain device drivers and other service providing software modules which must be permanently active. See also description of ISA applications above.
ESA applications are stored on the flash disk of the ID1021.

HTTP client: PC or workstation running HTTP client software, usually in the form of a web browser. (e.g. Microsoft Internet Explorer of Netscape Navigator)

Embedded HTTP server: Central aspect in the architecture is the Embedded HTTP server. HTTP clients can call upon the HTTP server for retrieving web pages and picture files. The HTTP server calls the EFS driver for retrieving those files from the flash disk of the ID1021 and transmits the file data using the HTTP protocol to the web client. If a HTML page contains a reference to an ISA application then the HTTP server will start the ISA application for retrieval of run-time data and insertion of run-time data into the HTTP data stream to the HTTP client. As soon as all of the data for a HTTP page is transmitted to the client the ISA application is stopped.

2.2 HTTP sample sessions

2.2.1 Static HTML

The following example illustrates a HTTP session for a static HTML page (e.g. static.htm), i.e. a page that contains no items that must be filled at run-time in by the HTTP server.

Step	HTTP client side (web browser)	ID1021 side
1	Requests file static.htm (with HTTP 'Get' or 'Post' command)	
2		HTTP server starts the default.isa application as the requested file has the .htm extension.
3		The default.isa application scans the flash disk for password.txt file. If such a file exists, then it is read and a password prompt is presented to the HTTP client first. If none exists, then the requested HTML file (static.htm) is transmitted to the HTTP client.
4		The default.isa application is ended when transmission of HTTP data is complete.
5	Displays the received HTTP data (contents of the static.htm)	

Table 3: HTTP session for static HTML page

In this example the HTTP client receives the exact contents of the static.htm file as it is stored on the flash disk of the ID1021. Not a byte is changed. That's why we call it static.

2.2.2 Dynamic HTML

The following table illustrates a HTTP session for a dynamic HTML page (e.g. temp.htm), i.e. a page that contains a temperature items that must be filled at run-time in by the HTTP server. It is to contain the actual temperature of the OEM device, i.e. the temperature at the moment in time the HTTP server was called by the client.

Step	HTTP client side (web browser)	ID1021 side
0		The oem.esa application communicates with the OEM device and receives a new temperature value every second.
1	Requests file temp.isa (with HTTP 'Get' or 'Post' command)	
2		HTTP server starts the temp.isa application.
3		The temp.isa application communicates with oem.isa application and retrieves the latest OEM device temperature value.
4		The temp.isa application opens the temp.htm file. It calculates the new (dynamic) contents size and reports this size to the HTTP server. (as it is about to replace parts of the original HTML contents with run-time temperature data - the size of the content will therefore be different from that of the static temp.htm contents) The temp.isa also hands a buffer to the HTTP server that contains the actual static content from the temp.htm file.
5		The HTTP server transmits a HTTP header to the

		HTTP client that contains the new contents size.
6		The HTTP server reads content data from the buffer and parses the data. If it encounters an ASP tag in the data then it calls the temp.isa to replace the ASP tag with the run-time temperature value. All other content data is transmitted transparently to the HTTP client.
7		The temp.isa application is ended when transmission of HTTP data is complete.
8	Displays the received HTTP data (= contents of the temp.htm – with ASP tag replaced by latest temperature value)	

Table 4: HTTP session for dynamic HTML page

Note that in this example the HTTP client does not request the temp.htm file itself but the temp.isa application that goes with this HTML file. Note that as a result of this the default.isa application is not called by the HTTP server. Instead the temp.isa application is called. The temp.isa on its turn opens the temp.htm file and passes it contents to the HTTP server.

An ASP tag is a special HTML tag. (refer to [HTML] for more details about tags in general or the ASP tag especially) Normally with the ID1021 it has the format

<%Txx%>

where *xx* is a unique decimal tag number.
The unique tag number is used by the ISA application to identify which tag was encountered by the HTTP server and with what run-time value it is to be replaced. In the example above there was only one ASP tag – the one for the temperature value – but in other HTML pages their might by more than one ASP tag.

2.2.3 Passing parameters to an ID1021 application

The table below illustrates another HTTP session example. This time the user clicks on a button in the fan.htm page to start the cooling fan that is integrated in the OEM device. The fan.htm page also contains an indicator representing the current status of the cooling fan. (on/off)

Step	**HTTP client side (web browser)**	ID1021 **side**
0		The oem.esa application communicates with the OEM device and receives a new temperature sample every second.
1	User clicks on 'start fan' button. The web browser builds a HTTP query string for the button and requests file fan.isa (with HTTP 'Get' or 'Post' command) The HTTP command includes the query string)	
2		HTTP server starts the fan.isa application. It hands the parameters from the query string to the fan.isa application.
3		The fan.isa checks the parameters and determines that the cooling fan is to be switched on. It calls

		the oem.esa to activate the cooling fan. It also updates its own administration for the change of state of the cooling fan.
4		The fan.isa application opens the fan.htm file. It calculates the new (dynamic) contents size and reports this size to the HTTP server. (as it is about to replace parts of the original HTML contents with the new cooling fan state - the size of the content will therefore be different from that of the static fan.htm contents) The fan.isa also hands a buffer to the HTTP server that contains the actual static content from the fan.htm file.
5		The HTTP server transmits a HTTP header to the HTTP client that contains the new contents size.
6		The HTTP server reads content data from the buffer and parses the data. If it encounters an ASP tag in the data then it calls the fan.isa to replace the ASP tag with the new cooling fan state. All other content data is transmitted transparently to the HTTP client.
7		The fan.isa application is ended when transmission of HTTP data is complete.
8	Displays the received HTTP data (= contents of the fan.htm – with ASP tag replaced by latest cooling fan state)	

Table 5: HTTP session for dynamic HTML page

Note that this example is only different from the previous one in the sense that now a parameter is transferred from the HTTP client to the HTTP server. The HTTP server hands this parameter to the ISA application, which then takes the appropriate action. (calling the oem.esa to switch on the cooling fan)

HTTP query strings in general take the form of

URL?parameter1=value1&parameter2=value2& parameter3=value3

where

- *URL* is the location of the ISA application, in our example: /fan.isa
- *parameter1* is the name of the first parameter to be transferred
- *value1* is the new value for *parameter1*
- *parameter2* is the name of the first parameter to be transferred
- *value2* is the new value for *parameter2*

etc.

Refer to [HTTP] for more detailed information about URLs, HTTP query strings and parameters.

In our example the HTTP query string (with just one parameter) could have been like this:

/fan.isa?coolingfan=on

For building the query string the web browser uses the names and values for the parameters that are mentioned in the HTML source file. For the button in our example the fan.htm file would then include a HTML fragment like this:

<input type="submit" value="on" name="coolingfan">

As the fan.isa application uses the name and value of the parameter to determine what action should be taken, one can understand that is crucially important these names and values must not be changed when adapting the HTML files. (for example for translation of the HTML page to the Dutch language)

In fact the names and values of the query parameters form the interface between the user interface (HTML) part and the OEM device interface part. (ISA, ESA applications)

2.3 Start page

If no filename is specified in the URL, e.g. when only the IP address of the ID1021 is entered in URL-window of a web browser, then the ID1021 HTTP server will call the default.isa application for presenting the default start page. This page is the page with the name 'index.htm'.
So for displaying the start page it is required that at least the default.isa and the index.htm files are present on the flash disk of the ID1021.

3 Customizing HTML

Now that we know about the basic architecture of the ID1021 HTTP server we can start customizing the HTML pages for it.

Usually the main reasons for customizing the HTML pages are:

- Translation of the user interface into another (foreign) language (internationalization)
- Re-styling the user interface to match the company's house style.

For both situations we assume there is a working set of existing HTML pages and ISA applications.

3.1 Internationalization

With internationalization is meant setting up the HTML files for different languages. The java script language that can be used in HTML pages for the ID1021 support features that enable detection of the type and version of the web browser the client is using as well as his nationality/language. Based on this it should theoretically be possible to use different set of HTML pages for user with different languages. (run-time language switching) We have not tested/implemented this yet at Necoso.

Other aspects to keep in mind when internationalizing your web pages:

- The default encoding or character set that is used for the web page. Some web browser support auto-detect of the required character encoding, others do not. We usually 'encode' our webpage using the ISO 8859-1 character set. The ISO-8859-1 character set is also referred to as the 'Western European' character set or the 'Windows-1252' character set. The first 128 characters in the ISO-8859-1 character set match the ASCII character set. (HTML: '*charset=windows-1252*') This character set includes diacritical characters most western (European) languages and will do for most ID1021 applications used in the United States and Europe.
- Texts inserted in the HTTP data stream by the ISA applications them selves (in dynamic HTML pages) must be translated by changing the ISA applications themselves or extending them for support of multiple languages by means of a query parameter that indicates the wanted user interface language. Typical candidates for such solutions are ISA generated prompts, error messages, etc.

3.2 Adding new HTML files

New static HTML files can be added with out any problems. (e.g. for adding your own HTML help pages)
References to these new pages can be included in existing HTML pages the usual way, i.e. by including direct hyperlinks to the new pages. The same goes for adding new (static) picture files (.GIF, .JPG, etc)

New dynamic pages can only be added when the corresponding ISA applications are also added or an existing ISA is used. The latter option requires detailed information about the ISA, its parameters and its parameter values.

3.3 Adapting existing HTML files

3.3.1 ASP tags

When adapting the HTML files care should be taken to not change the ASP tags that are used by the ISA applications. ASP tags for the ID1021 have the following format:

<%Tnn%>

where *nn* stands for a 2 digit decimal tag number.
IMPORTANT: ASP tags must not be removed from the HTML files, nor must new ones be added.
The order in which the ASP tags appear in the HTML file is not important, though. They may be moved around.

3.3.2 References to ISA applications

IMPORTANT: Don't change references to ISA applications (e.g. in FORM tags).
Don't change GET commands into POST commands or vice versa.

3.3.3 Other HTML items

Other HTML items that must not be changed are the names and values for user interface items in OPTION, SELECT or INPUT tags. As we have seen in paragraph 2.2.3 the ISA applications depend on these, so don't change them!

3.4 Testing the adapted HTML files

Some hints for testing the adapted HTML files:

- Always keep a copy of the original HTML files. So you can do a file comparison if things go wrong or you encounter unexpected behavior.

- Test with more then one web browser. Experience has learned that Microsoft Internet Explorer is less critical about HTML errors like open tags than for example the Netscape Navigator. Pages that look OK with Microsoft Explorer may give unexpected results with Netscape Navigator.

- Test the HTML pages at different screen resolutions. Not all users may use the same resolution as you have on your development PC.

- If you customized your web pages for a foreign language, then test the language specific and or OEM device specific characters. (ASCII/ANSI characters > 127) Especially where the OEM device is a device that uses the web based user-interface input for means of its own. For example if the OEM device is a LED or LCD display, then test that the characters that are input at the web interface are displayed as expected on the display itself. Typical test case is here the 'euro sign' (€) for example.

3.5 Other hints

- Keep the HTML files lean & mean. Remove any superfluous comments, tags, spacing and other items that are not absolutely necessary. Remember that the ID1021 has only limited space for storing applications and web content. Also a user interface based on large HTML and picture files will decrease overall performance and will make a 'sluggish' impression on the user.

- HTML pages that contain pictures that need to be loaded every time the user requests the page, can be cached on the client side by using a meta tag in the header of the HTML file.

  ```
  <meta http-equiv="expires" content="thu, 08 jun 2067 00:00:00 gmt">
  ```

  This tells the web browser that the page will only expire in the far future and that it does not have to retrieve it again from the ID1021 every next time the user request the same page. The web browser will load this page only once and 'remember' the contents every next time. Note that this only works for static pages and if no query string is used with the page.

- Automatic reloading of a HTML page can be realized using a meta tag in the header of the HTML file. For example: the following fragment causes the HTML to be reloaded automatically every 5 seconds:

  ```
  <meta http-equiv="refresh" content="5">
  ```

  Alternatively java script could also be used to realize similar effects.

- By default the HTTP server of the ID1021 will present the index.htm file if no file name is specified in the URL in the web browser, see paragraph 2.3. The index.htm is a static web page and can therefore not contain any run-time information. What if you want a dynamic starting page that does contain run-time information? (i.e. if I want my temp.isa application to be called as starting page) There are 2 ways of realizing this:

  1) By writing your own default.isa application. Rather cumbersome.
  2) By using an index.htm with contents like this:

  ```
  <html>

  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
  <meta http-equiv="refresh" content="0;URL=temp.isa">

  <title>Index</title>
  </head>

  <body>

  </body>

  </html>
  ```

The second meta line with the "refresh" keyword in this index.htm will force the web browser to automatically load the temp.isa application 0 seconds after loading of the index.htm itself is complete.