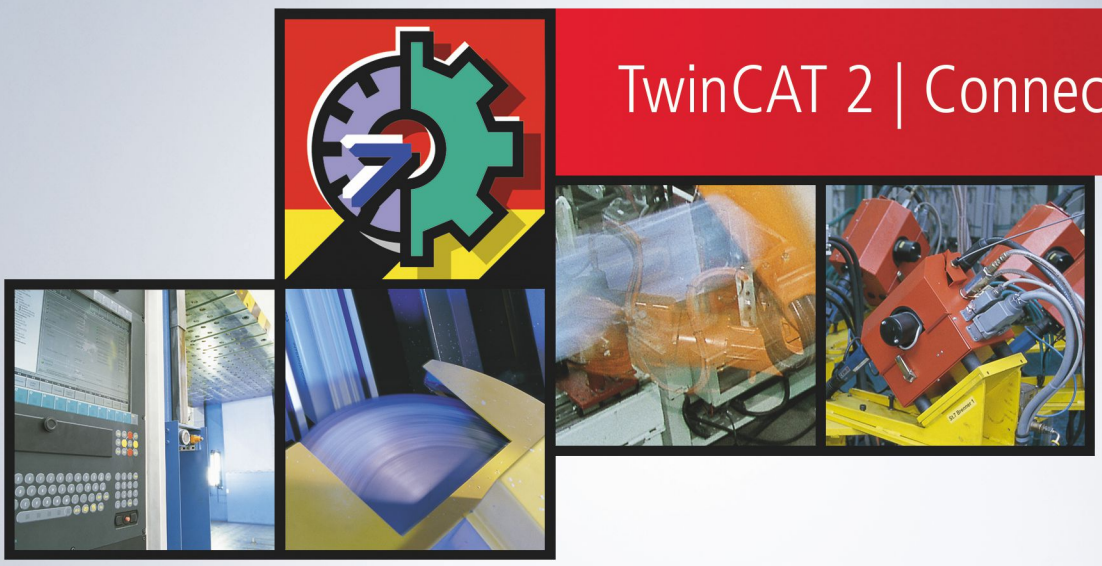
BECKHOFF New Automation Technology

Manual | EN

TX1000

TwinCAT 2 | ADS WebService



2023-08-11 | Version: 1.2

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

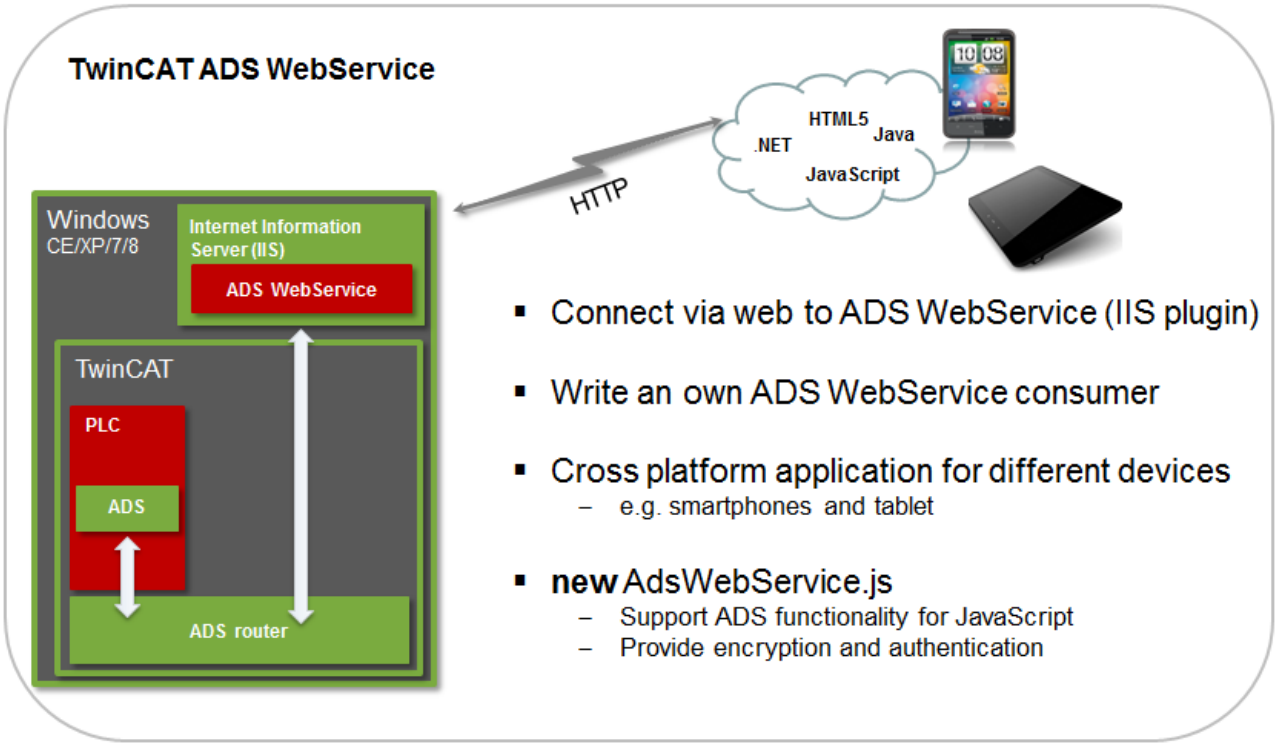
To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Introduction

In this document the **"ADS Communication via WebService"** is described.

The ADS-WebService enables the possibility to set up an ADS communication via HTTP through a firewall.

Now it is possible to use an ADS communication via internet for diagnosis and visualization.



Prerequisites

- PC with TwinCAT on Windows XP/7/8 or Windows CE System with IIS (Microsoft Internet Information Server)

Further articles

- Configuration for Windows CE [▶ 19]
- Configuration for Windows XP [▶ 13]
- ADS Webservice samples [▶ 55]

3 Installation

3.1 Install TwinCAT ADS Webservice on a PC (here: Windows 7)

This chapter describes the IIS configuration required for the TwinCAT ADS Webservice running on Windows 7. (Please note: The configuration can differ for other Windows operating systems).

Requirements

The TwinCAT ADS Webservice is delivered by the default TwinCAT installation. This chapter describes how to configure Internet Information Service 7.

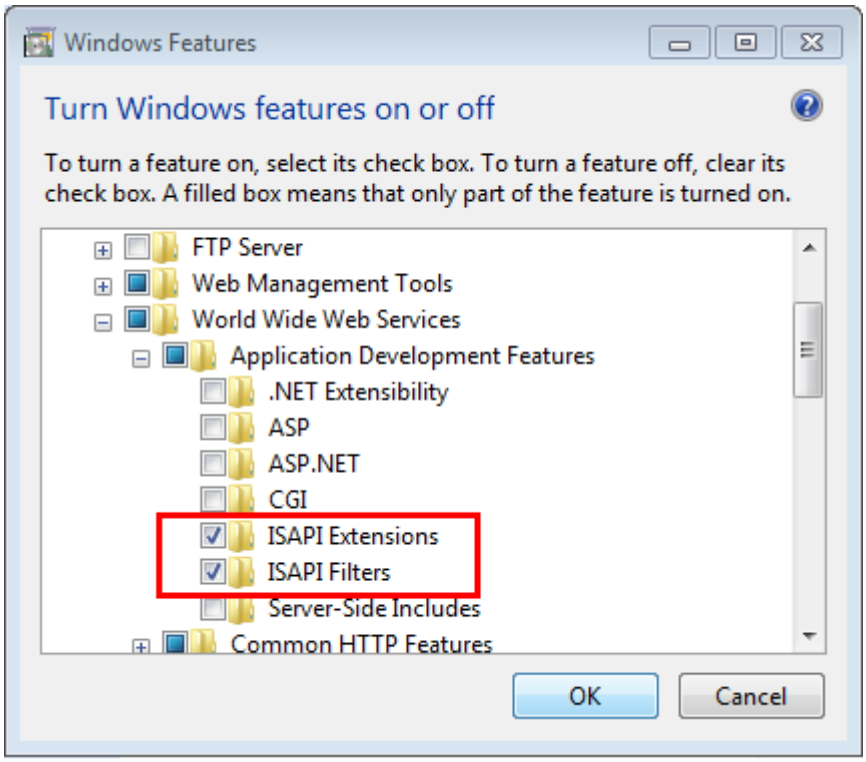
Step 1: Installing IIS7

By default, IIS7 is not part of the Windows 7 installation. Therefore you need to add this functionality manually. For more information see http://technet.microsoft.com/de-de/library/cc725762%28v=ws.10%29.aspx

Mandatory IIS7 extensions

When installing IIS7, the following extensions need to be activated:

- ISAPI Extensions
- ISAPI Filters

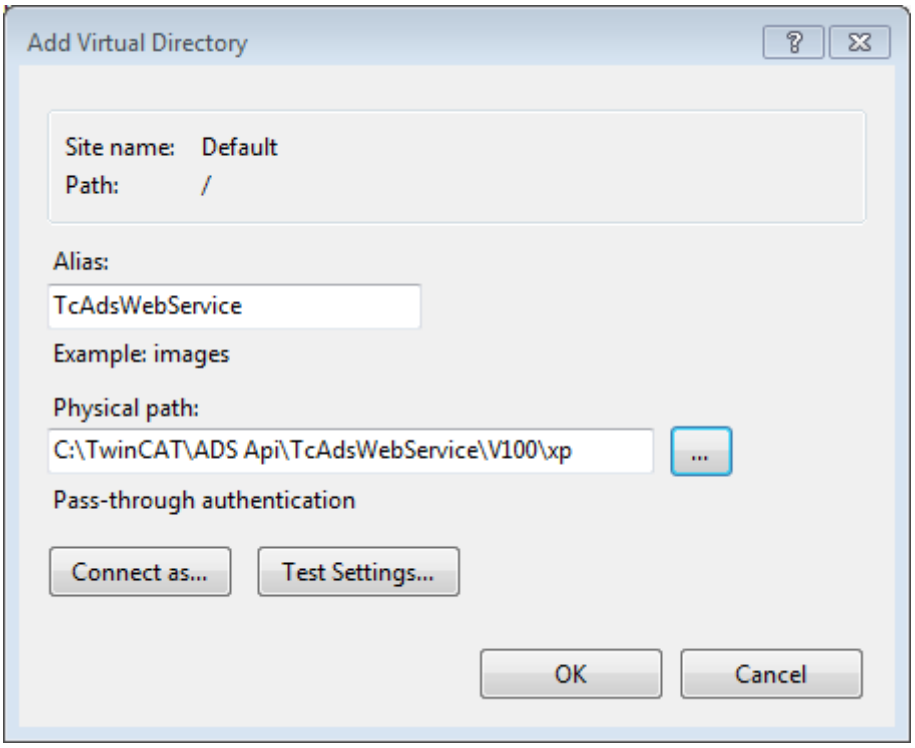


Step 2: Create "Virtual Directory" in IIS7 (Internet Information Service)

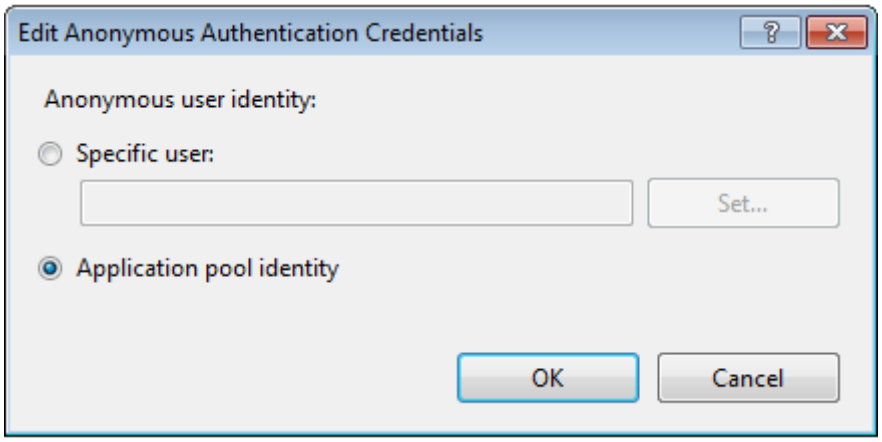
Usually, this step will be performed automatically by the Setup.

- Open "Internet Information Service (IIS) Manager" which can be found under "Control Panel\Administrative Tools\"
- Right click on "Default Web Site"
- Select "Add Virtual Directory..."

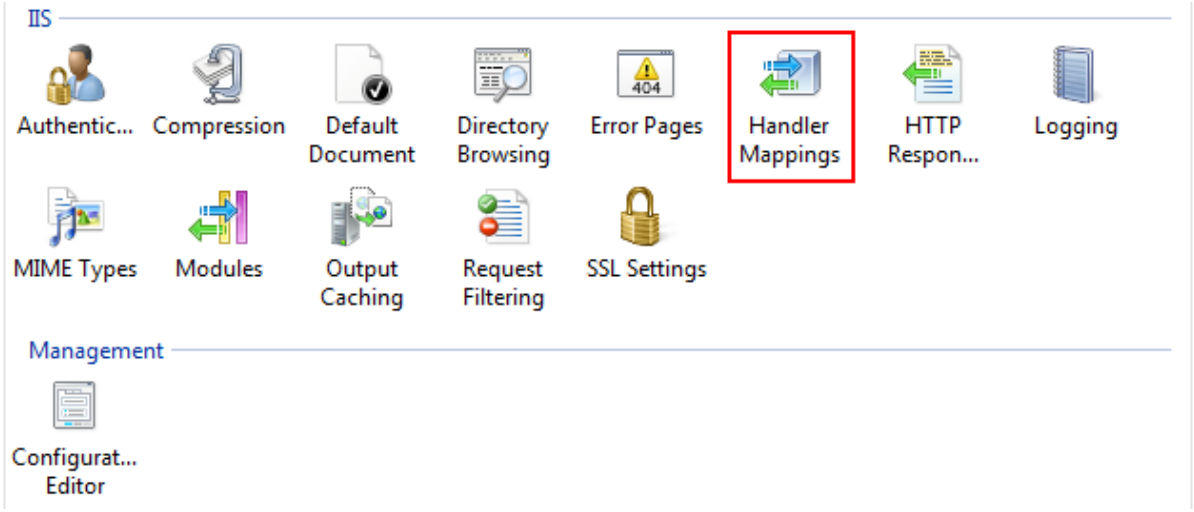
- Please enter the alias "TcAdsWebService" and the physical path to your TwinCAT ADS Webservice installation
  - For TwinCAT2 use C:\TwinCAT\ADS Api\TcAdsWebService\100\xp
- Click on "OK"



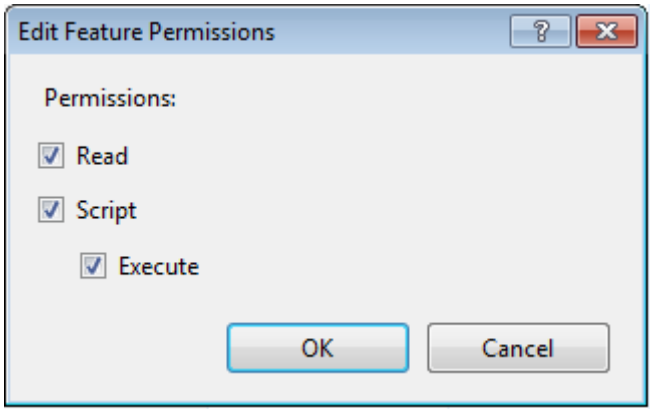
- Double-click the "Authentication" icon, select "Anonymous Authentication" and click on "Edit". Instead of specifying a user account, select the "Application pool identity" and click on "Ok".



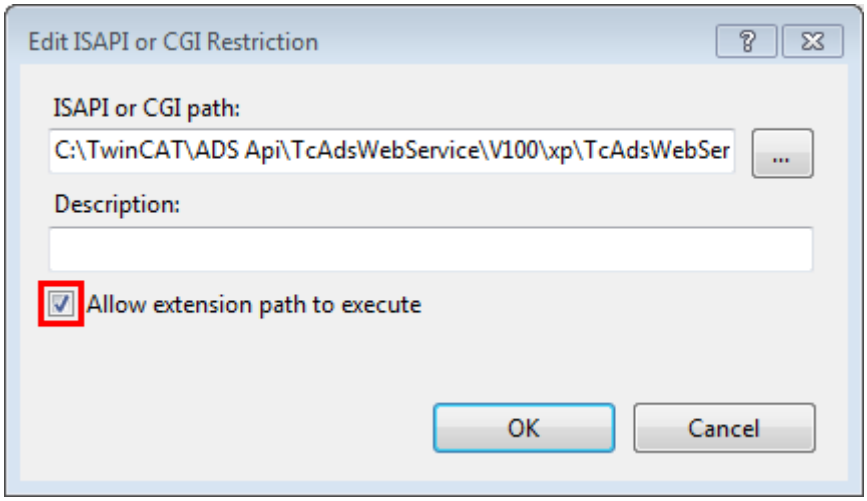
- Next, you need to set execute permissions on that virtual directory. Select the added directory and double-click on "Handler Mappings"



- Click on "Edit Feature Permissions" and select the "Execute" permission. Click on "OK".

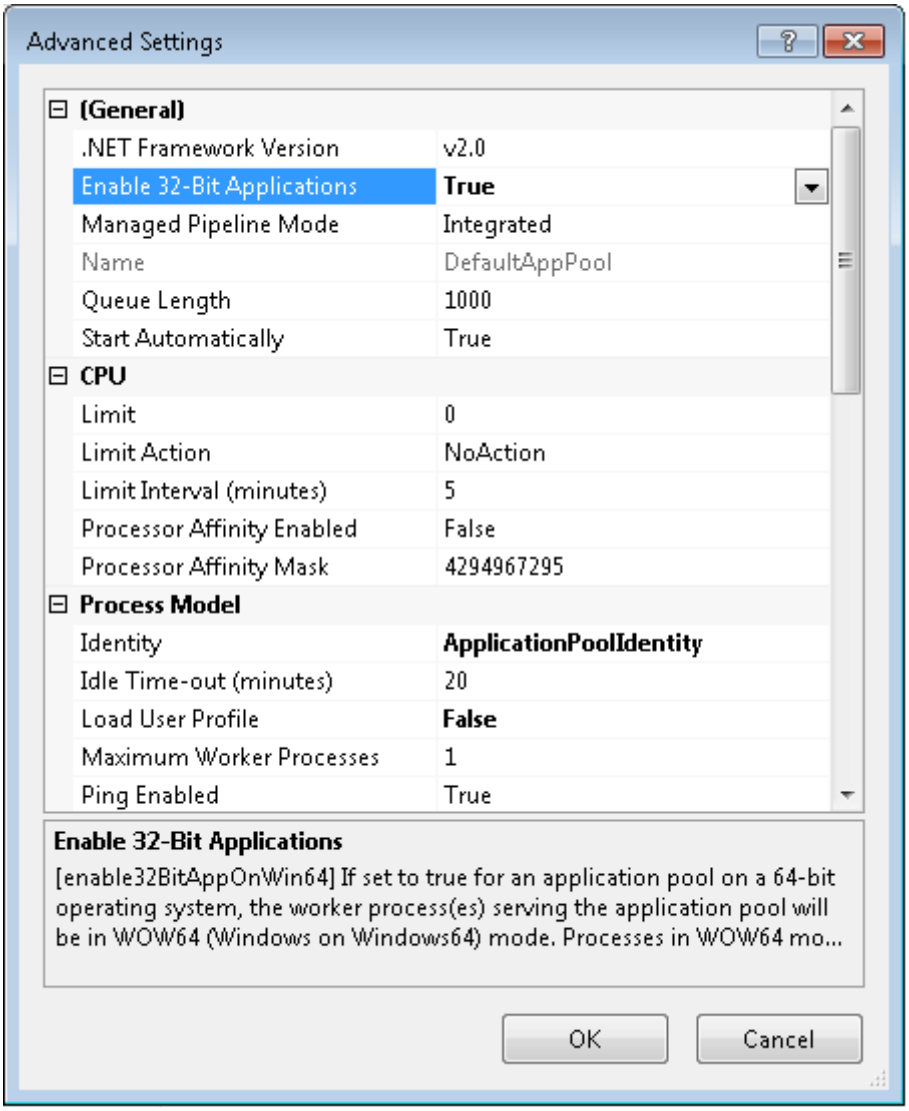


- As a next step you need to create an ISAPI allowance. Please select the root node in IIS Manager (named after your computer name) and then double-click "ISAPI and CGI Restrictions".
- Click on "Add" to create a new enabled extension. In the "ISAPI or CGI path" textbox, please specify the path to **TcAdsWebService.dll**:
  - Path for TwinCAT2: C:\TwinCAT\ADS Api\TcAdsWebService\100\xp
- Also select the checkbox "Allow extension path to execute".



- Restart your IIS.

- **If you use Windows 7 64-bit, you need to explicitly enable 32-bit ISAPI-DLLs in IIS. In this case, please perform the following steps.**
- Open "Internet Information Service (IIS) Manager" which can be found under "Control Panel\Administrative Tools\"
- Click on "Application Pools"
- Select the "DefaultAppPool" and click on "Advanced Settings..." from the Actions panel
- Set the entry "Enable 32-bit Applications" to "True", then click on "OK" to commit the changes



Step 3: Testing TwinCAT ADS Webservice configuration

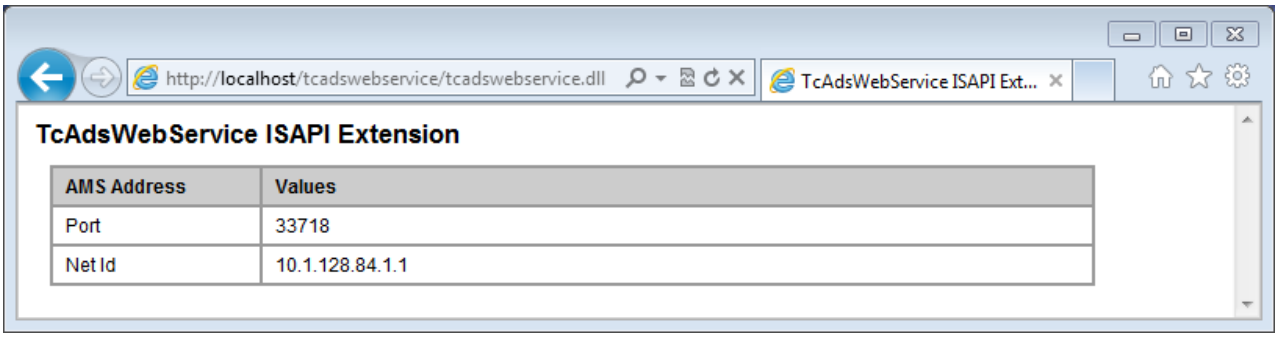
The URL of the TwinCAT ADS Webservice on the PC system can be accessed locally or from a remote computer. In both cases, open the web browser (e.g. Internet explorer) and enter the URL of the TwinCAT ADS Webservice, for example:

http://<ip-adress or name of PC device>/tcadswebservice/tcadswebservice.dll

Examples:

http://192.168.0.1/tcadswebservice/tcadswebservice.dll
http://localhost/tcadswebservice/tcadswebservice.dll

The TwinCAT ADS Webservice will reply with a status page containing the portnr and netid. If you see this page, the installation and configuration of the ADS Webservice has been successful.



In case of problems (like receiving no HTML status data), please check, if your system uses a proxy server. After deactivating the proxy and reloading the URL, the TwinCAT ADS Webservice should reply with the status info above.

3.2 Configuration of the TwinCAT ADS WebServices on a PC (in this example: Windows XP)

The required components for the ADS-WebService are comming with installation of :

- current TwinACT Installation (e.g. TwinCAT 2.10 Build 1246)
- a separate installation (without TwinCAT) "TwinCAT ADS Communication Library"

necessary files:

To set up the TwinCAT ADS WebService the following files are needed (can be found in "C:\TwinCAT\ADS Api\TcAdsWebService\V100\xp\"):

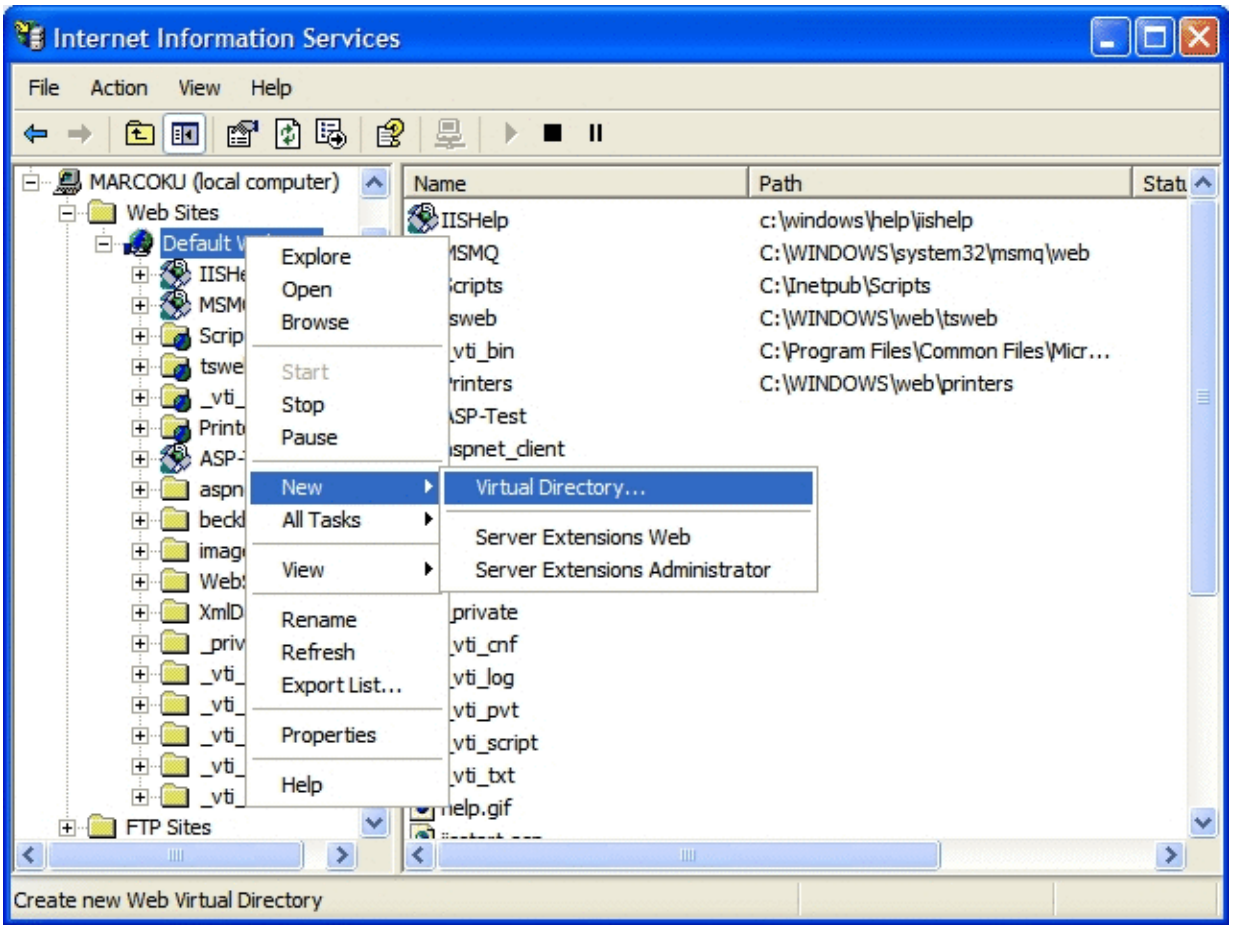
- TcAdsWebService.dll
- TcAdsSoap.dll
- TcAdsWebService.WSDL

This example explains the configuration on Windows XP. On other windows operation systems this configuration may differ in some aspects.

Creation of a new "Virtual Directory" in the IIS (Internet Information Service)

1. Open the "Internet Information Services" (in "Control Panel/Administrative Tools/").
2. Right-click in the tree-view on the righthand side on "Default Web Site".

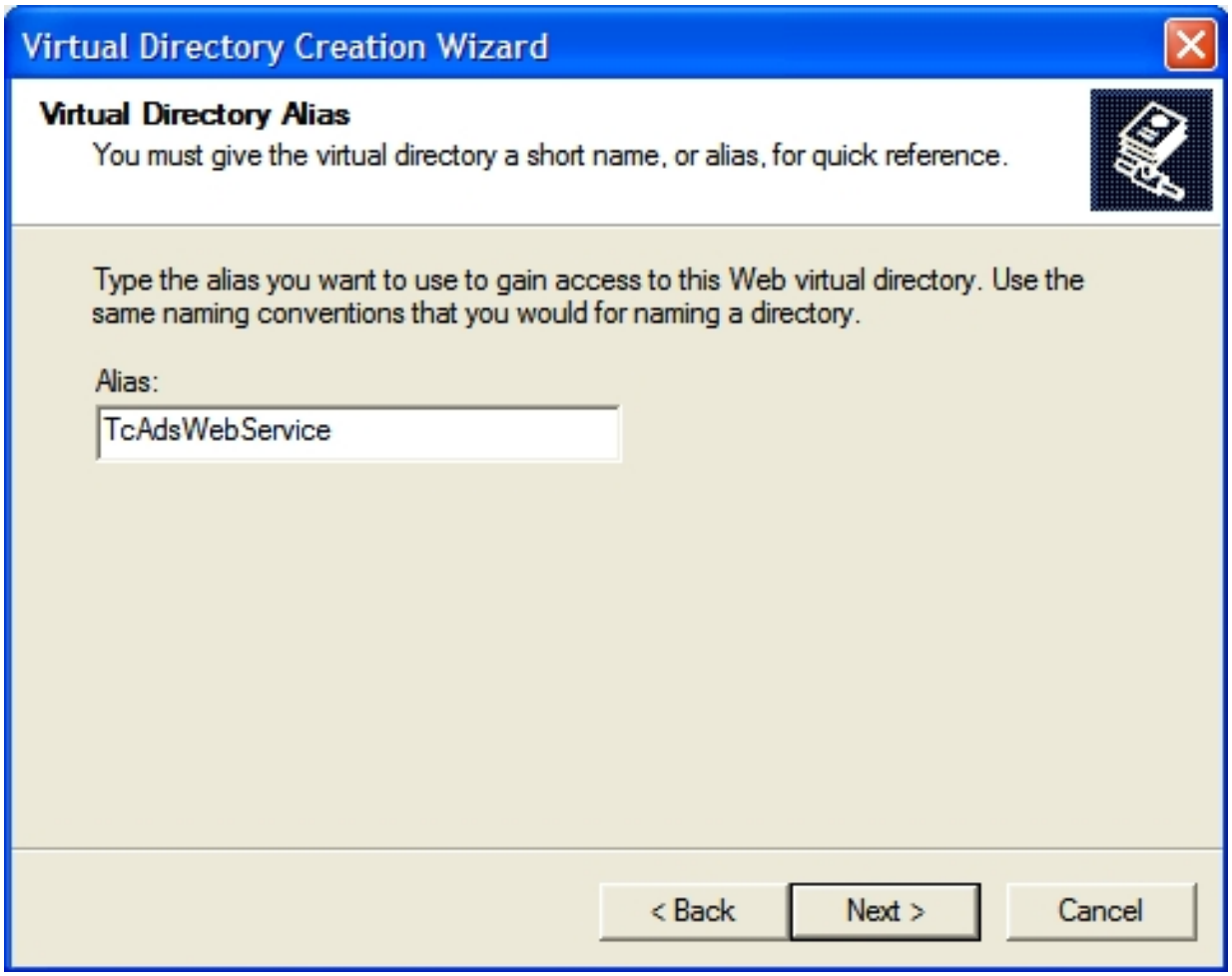
3. Select "New/Virtual Directory...".



4. Click "Next".


Virtual Directory Creation Wizard
Welcome to the Virtual Directory Creation Wizard
This wizard will help you create a new Virtual Directory on this Web site.
Click Next to continue.
< Back
Next >
Cancel

5. Insert "TcAdsWebService" as the alias and click "Next".



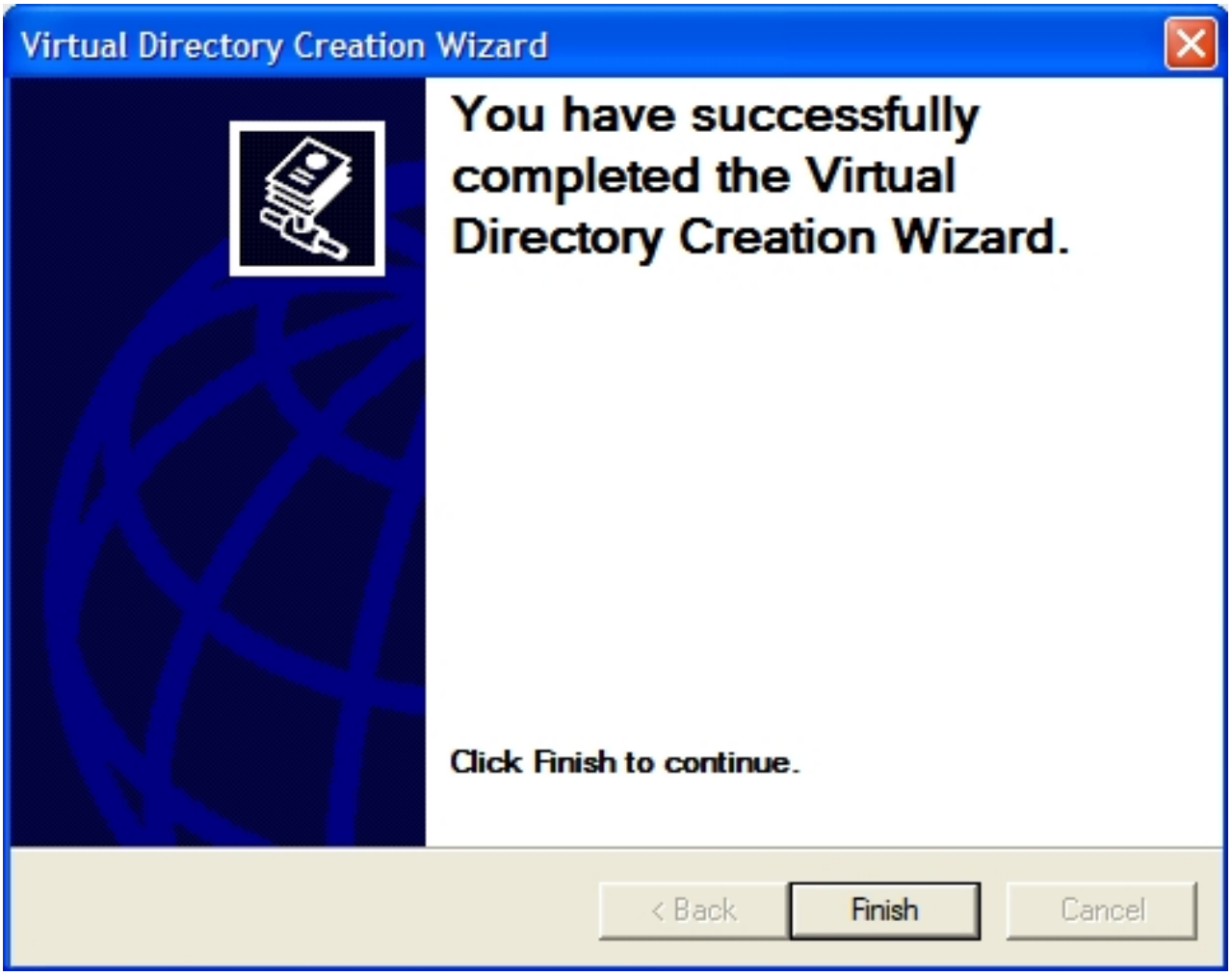
6. Insert the directory of the TwinCAT ADS Web Services (usually *"C:\TwinCAT\ADS Api\TcAdsWebService\V100\xp\"*). You can use the "Browse..."-Button to find this folder.

7. After that click the "Next"-button.

8. Check "Read", "Run scripts" and "Execute" and click "Next".

9. To complete the configuration of the TwinCAT ADS Web Service click "Finish".

⇨ Ready. The ADS WebService is now fully operational.



3.3 Configuration of the TwinCAT ADS WebServices on Windows CE

The necessary components for the ADS-Webservice installed with:

- current TwinACT Installation (e.g. TwinCAT 2.10 Build 1246)
- a seperat installation (without TwinCAT) "TwinCAT ADS Communication Library"

Necessary files:

To set up the TwinCAT ADS WebService the following files are needed (can be found in "C:\TwinCAT\ADS Api\TcAdsWebService\V100\ce\"):

- TcAdsWebService.dll
- TcAdsSoap.dll
- TcAdsWebService.WSDL

Create a new folder in "\hard disk\www\" with the name "TcAdsWebService" on your BECKHOFF-CE-Device (e.g. CX1001) and copy the 3 files mentioned above to this folder.

The configuration is completed.

3.4 Add virtual directory

TcAdsWebService.js only

This topic should be considered if you planning to use the JavaScript library TcAdsWebservice.js

The TcAdsWebService.js uses the TcAdsWebService. Therefore, you need to install the service first.

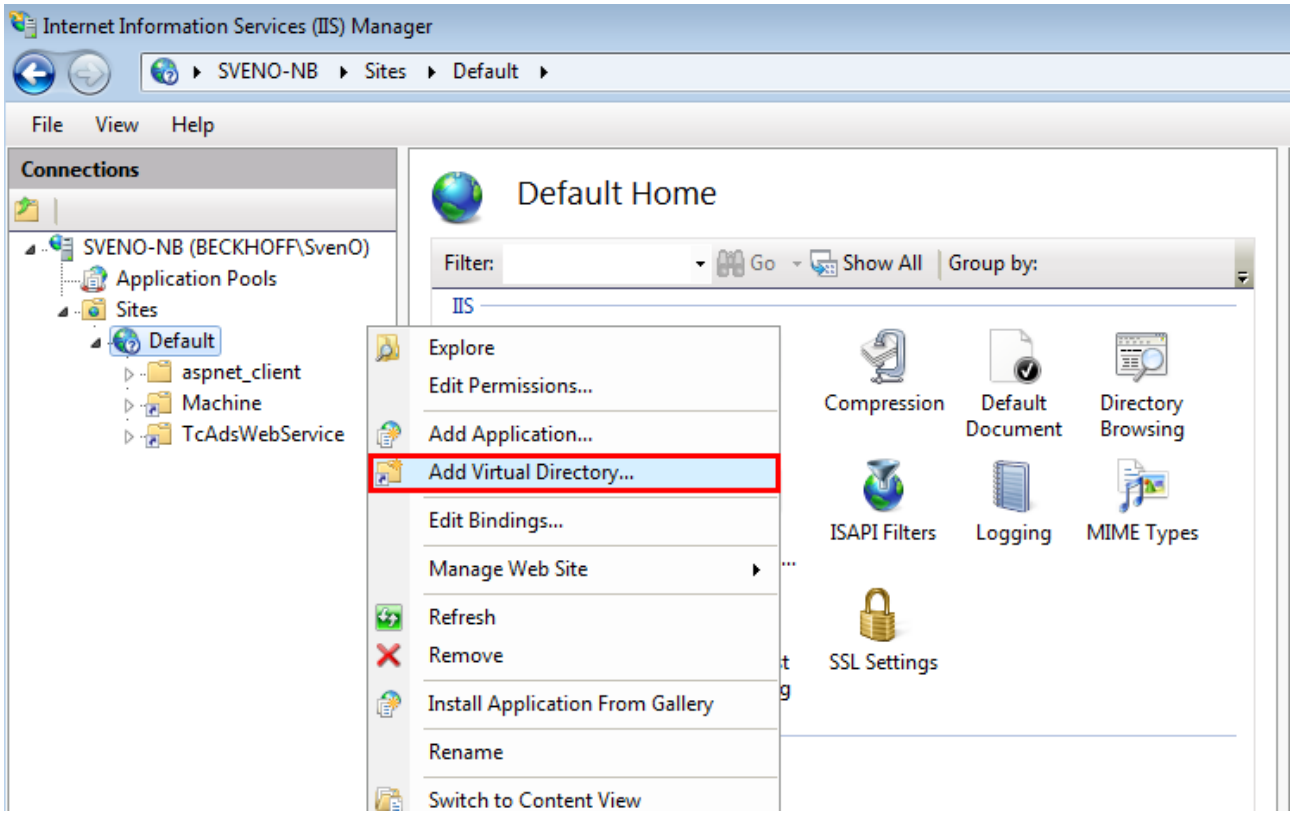
Please check our manual [▶ 8] and test the URL of the TcAdsWebService, for example:

http://<ip-adress or name of PC device>/tcadswebservice/tcadswebservice.dll

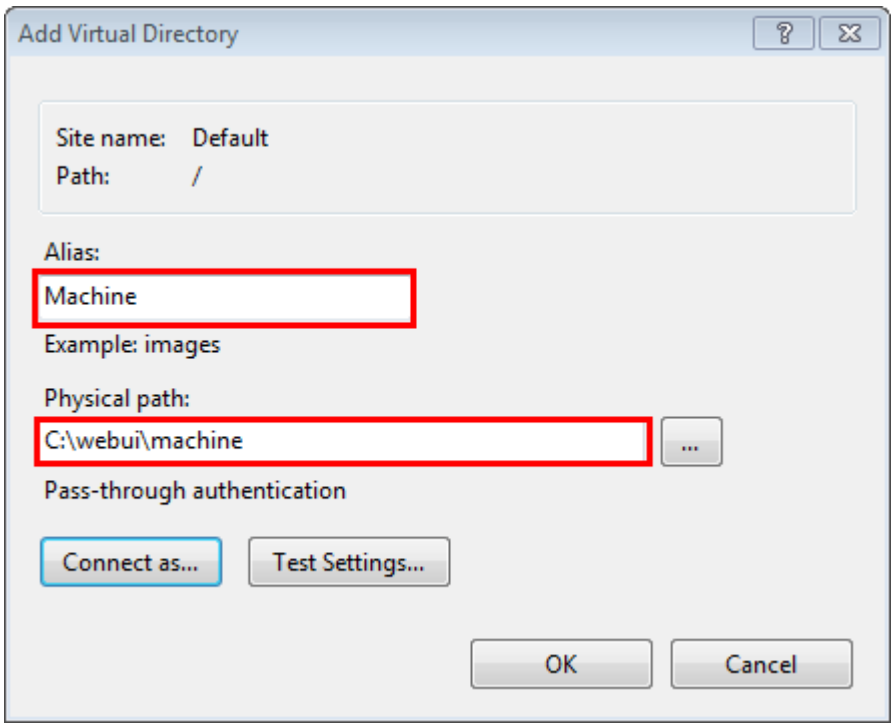
Examples:

http://192.168.0.1/tcadswebservice/tcadswebservice.dll
http://localhost/tcadswebservice/tcadswebservice.dll

For a new website a new virtual directory is needed for the internet information server (IIS). Switch to the configuration tool of the IIS (control panel/ adminstrative tools) and add a new virtual directory for your website.



Create an aliase for your URL and select the physical path of the HTML files:



In this example your website can be reached with following URL:
http://<ip-adress or name of PC device>/Machine

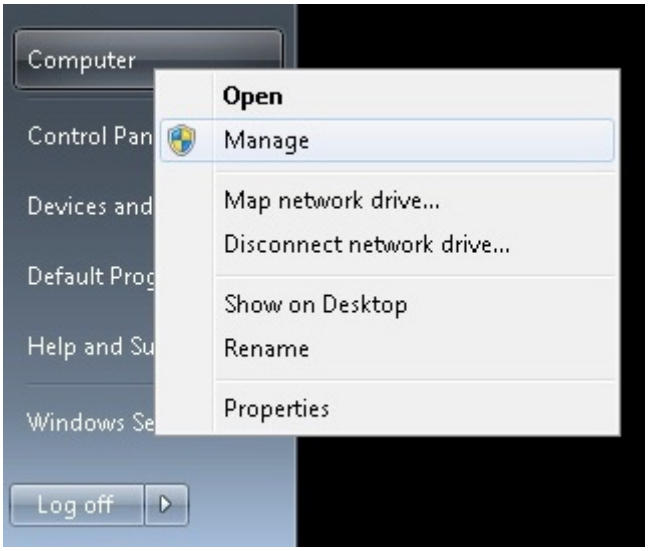
For Windows CE platforms create a subdirectory in /Hard Disk/WWW for your website.

4 Encryption/Authentification

4.1 Configuration of SSL/TLS and NTLM Authentification for the TwinCAT ADS WebServices on Windows 7

SSL/TLS Configuration

Rightclick on the Computer button in the Windows 7 Startmenu and press Manage to open the "Computer Management" window.



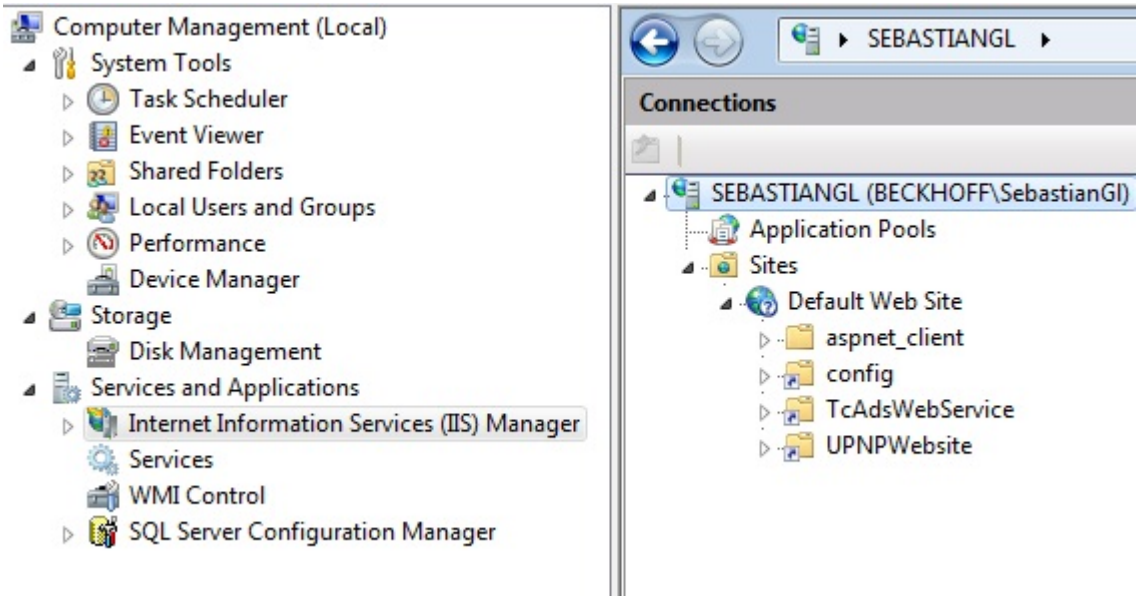
If you need a certificate for development or intranet use only, you can generate a self-signed certificate based on the computer name, directly in the Internet Information services by following the next steps here [▶ 22].

If you already have a server certificate which you want to use, you can follow the next steps here [▶ 22].

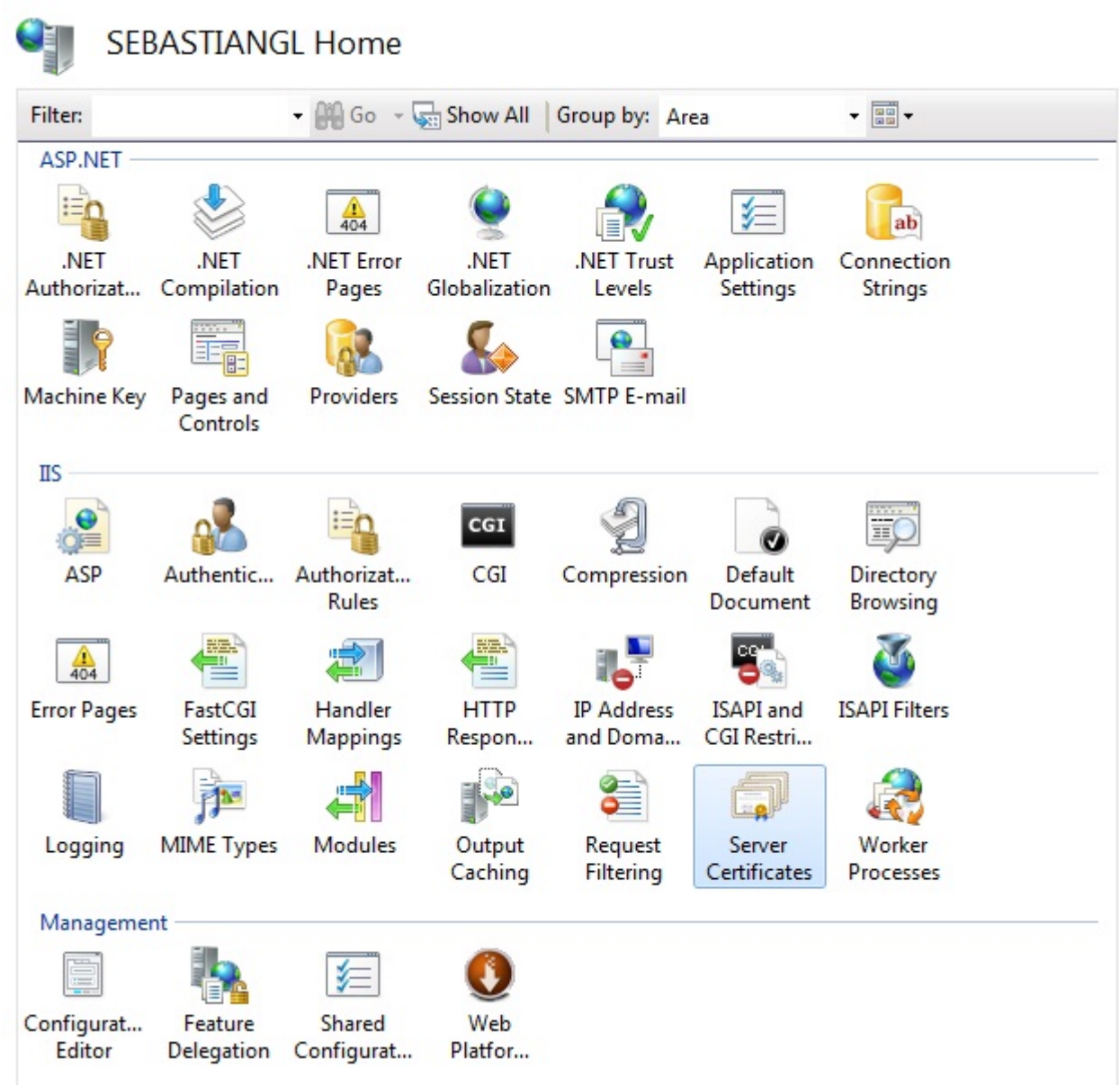
If you want to create a self-signed certificate for a domain in the world wide web, you can generate it with the SSLCert.exe [▶ 30].
If you have created a certificate with the SSLCert.exe you can follow the next steps here [▶ 22].

Creating a self signed certificate for development use

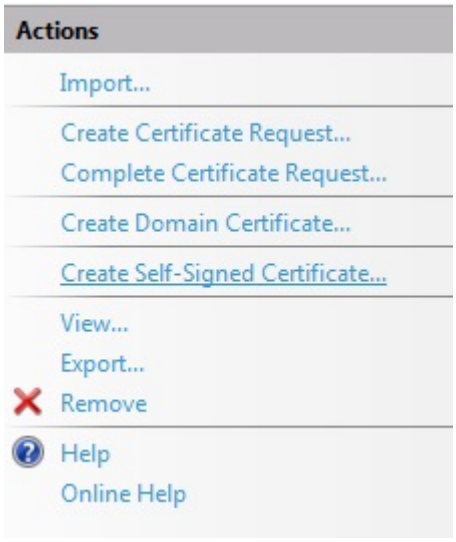
In the Computer Management Window navigate to "Services and Applications"->"Internet Information Servicess (IIS) Manager" and click on the root node of the Internet Information Services Manager navigation.



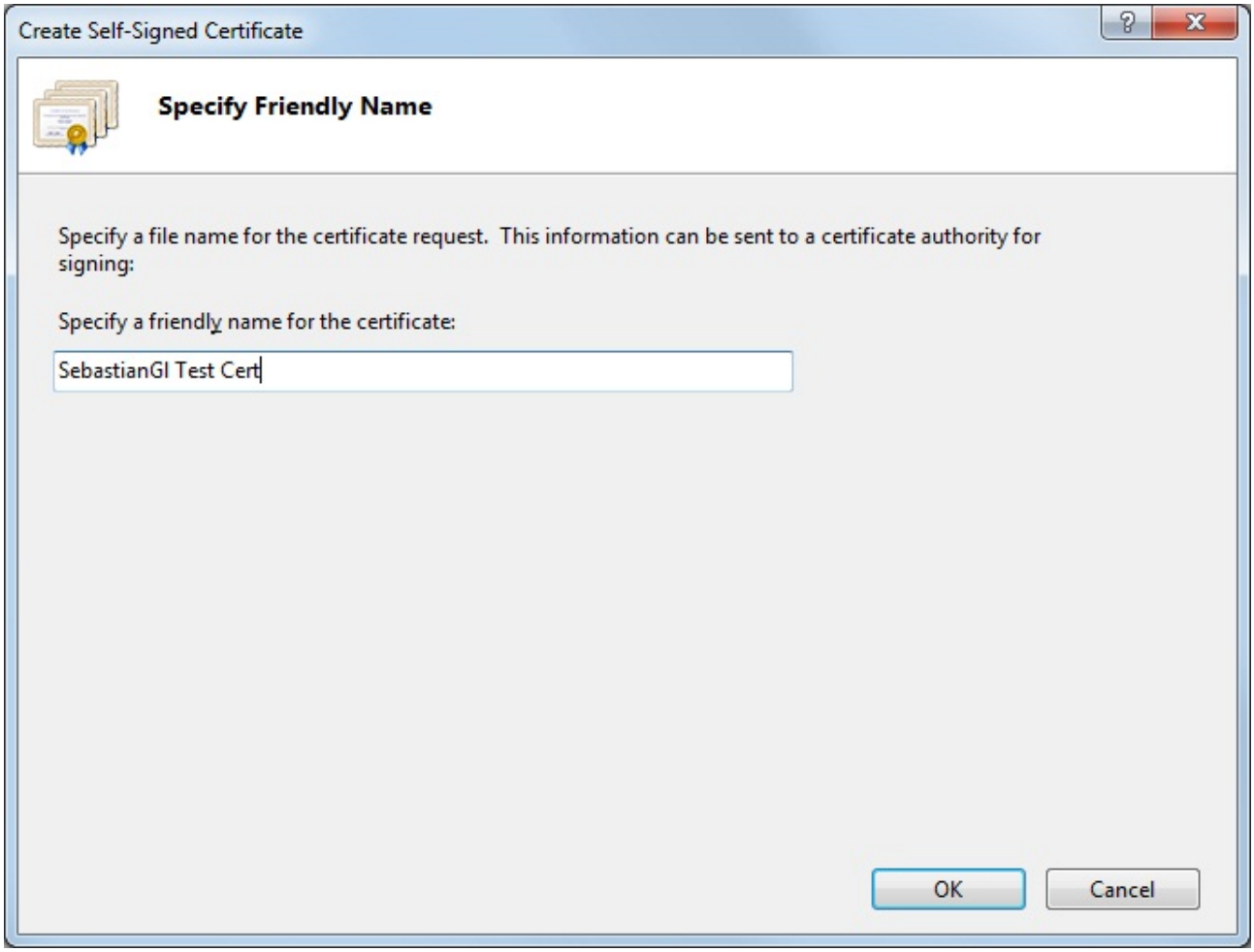
In the content pane of the Internet Information Services Manager choose the "Server Certificates" icon and double click on it.



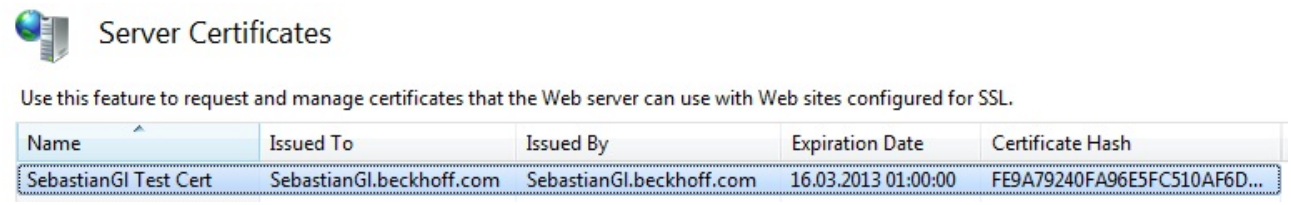
Click on the "Create Self-Signed Certificate..." menu entry in the Actions pane of the "Server Certificates" window.



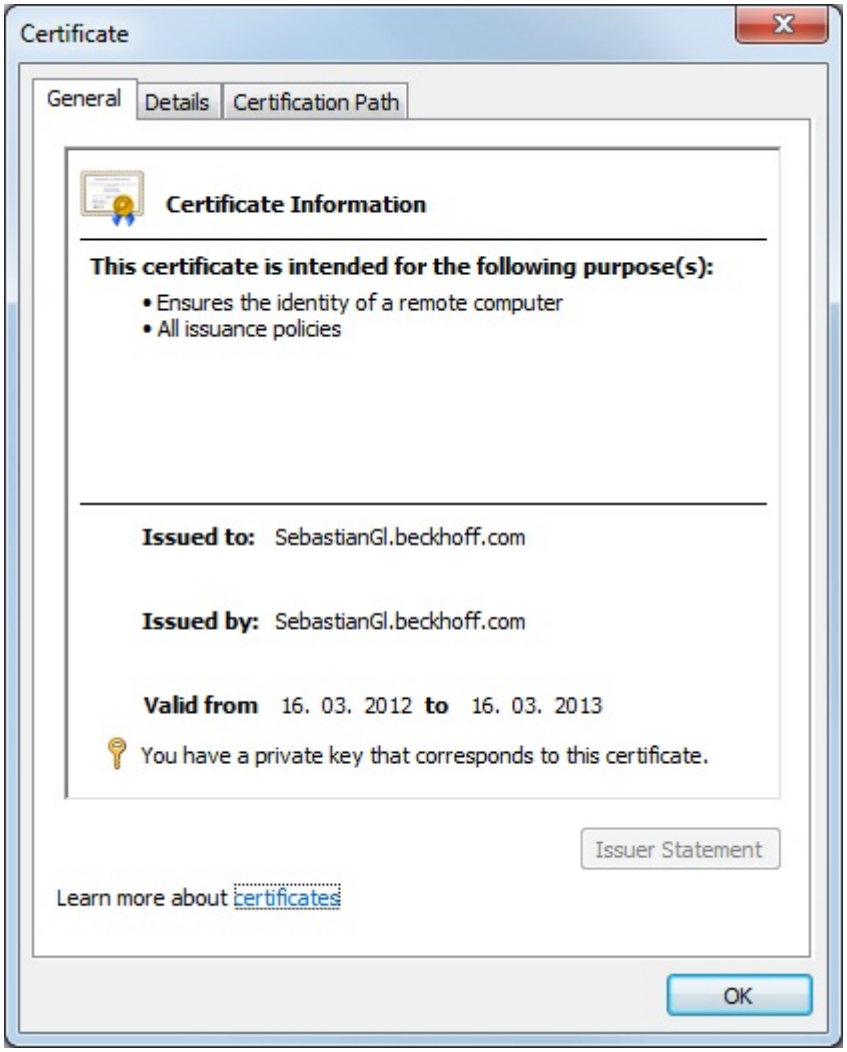
In the opened dialog you must specify a friendly name for your certificate. After that press the OK button to confirm.



The certificate was added to the certificate store of your computer and is displayed in the list of available Server certificates in the "Server certificates" window of the IIS Manager.

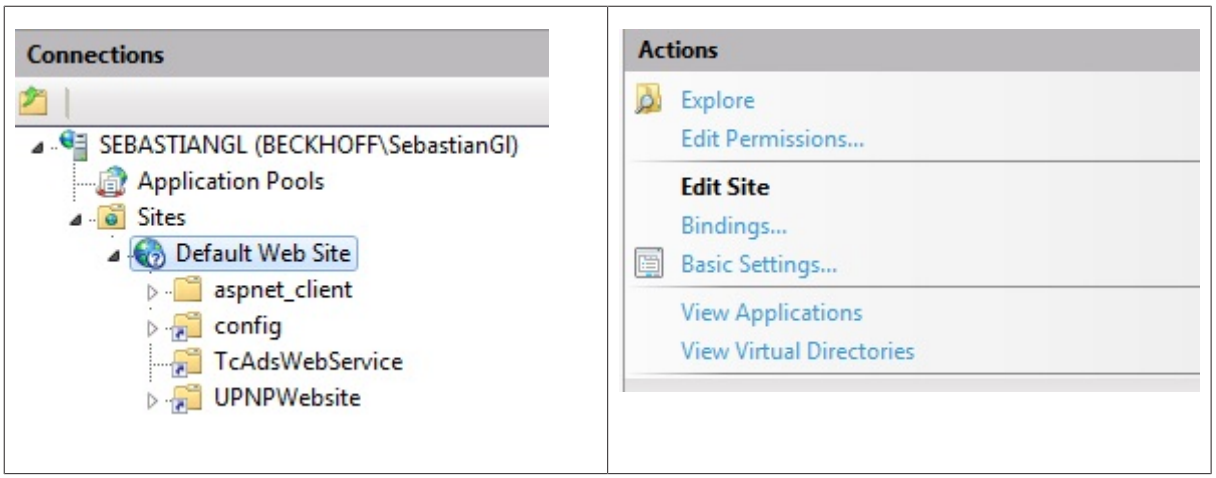


You can double click on the new entry to open the certificate information dialog for further information about the certificate.

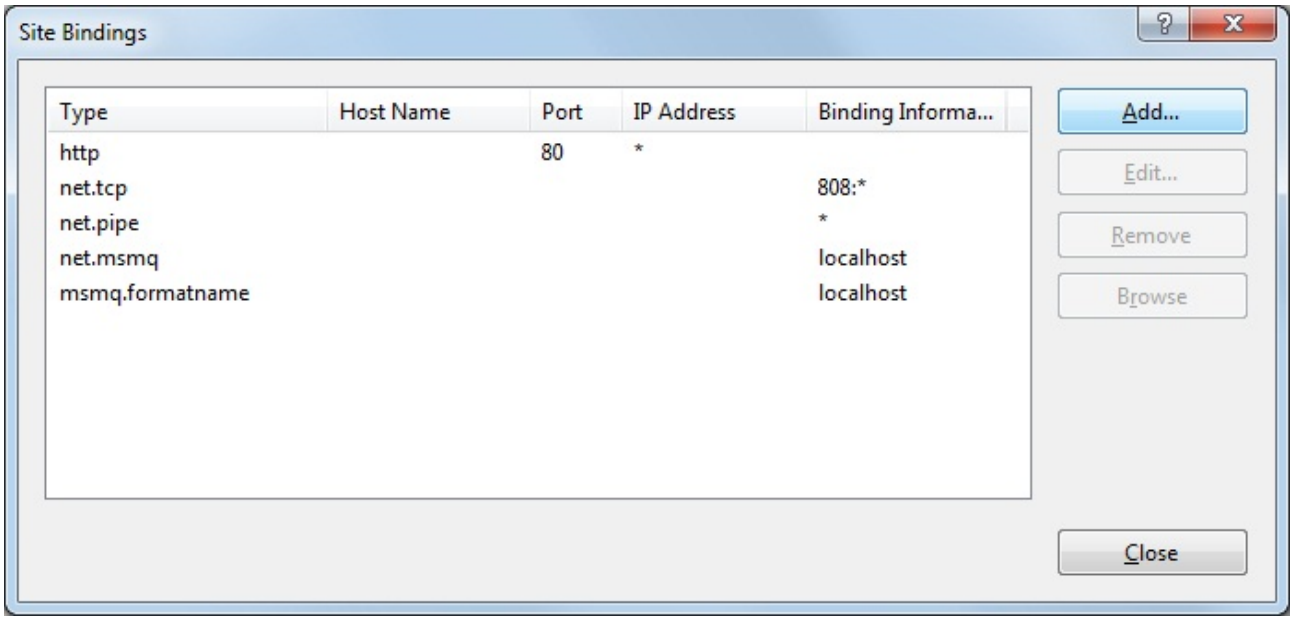


HTTPS Binding

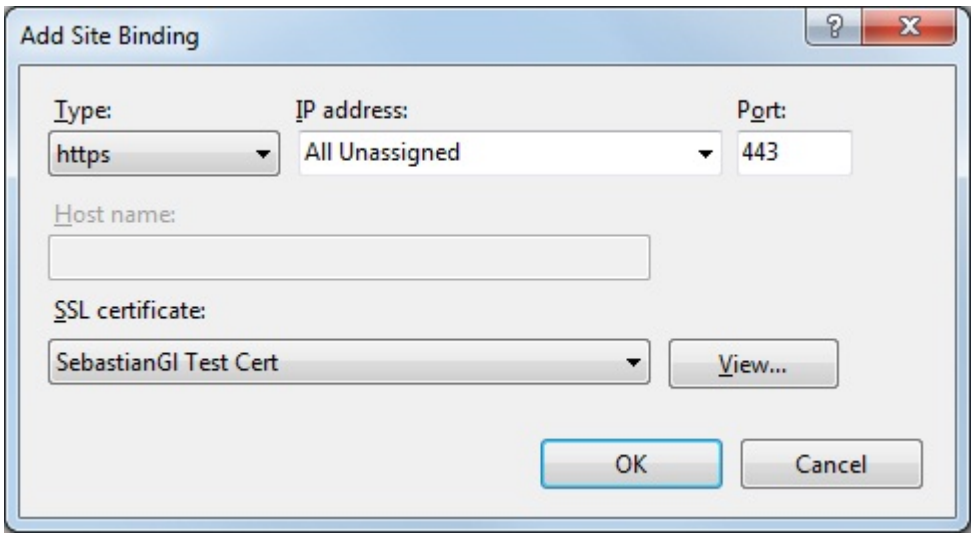
Now mark the website node in the Internet Information Services Manager navigation which contains the virtual directory for the TcAdsWebService and click on the "Bindings..." menu point in the appropriate Actions pane.



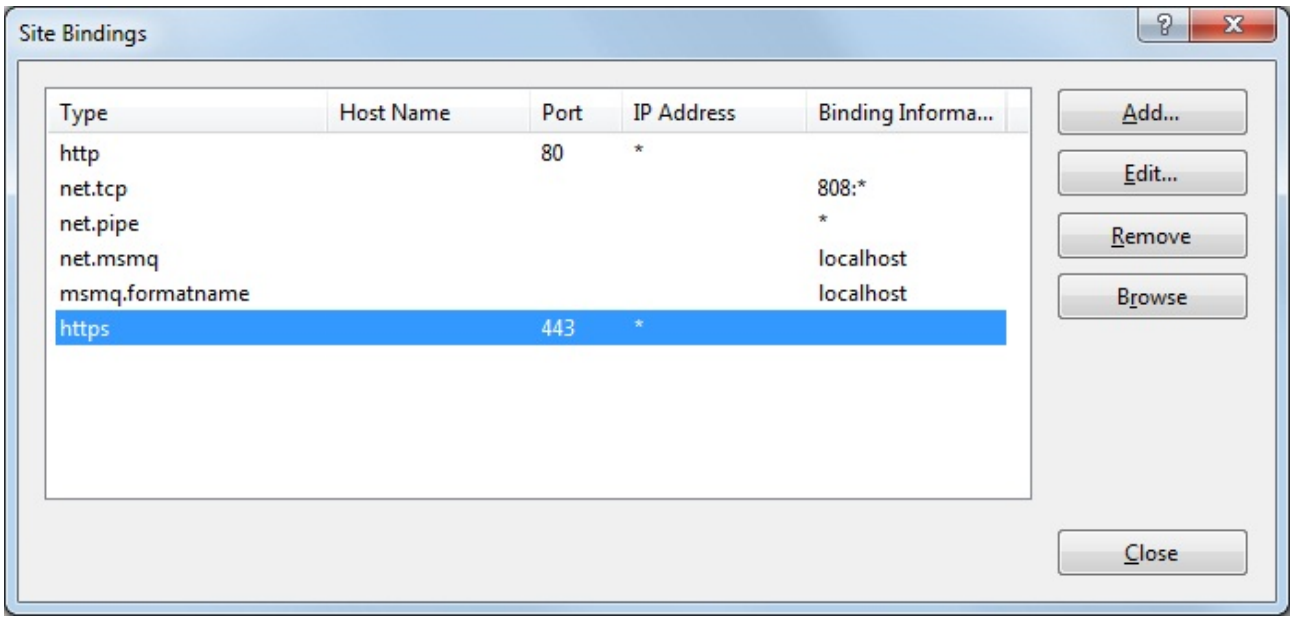
Now we must create a binding for the https protocol. Press the "Add" button to open the "Add Site Binding" dialog.



In the "Add Site Binding" dialog, choose the following values to create a binding for the https protocol.

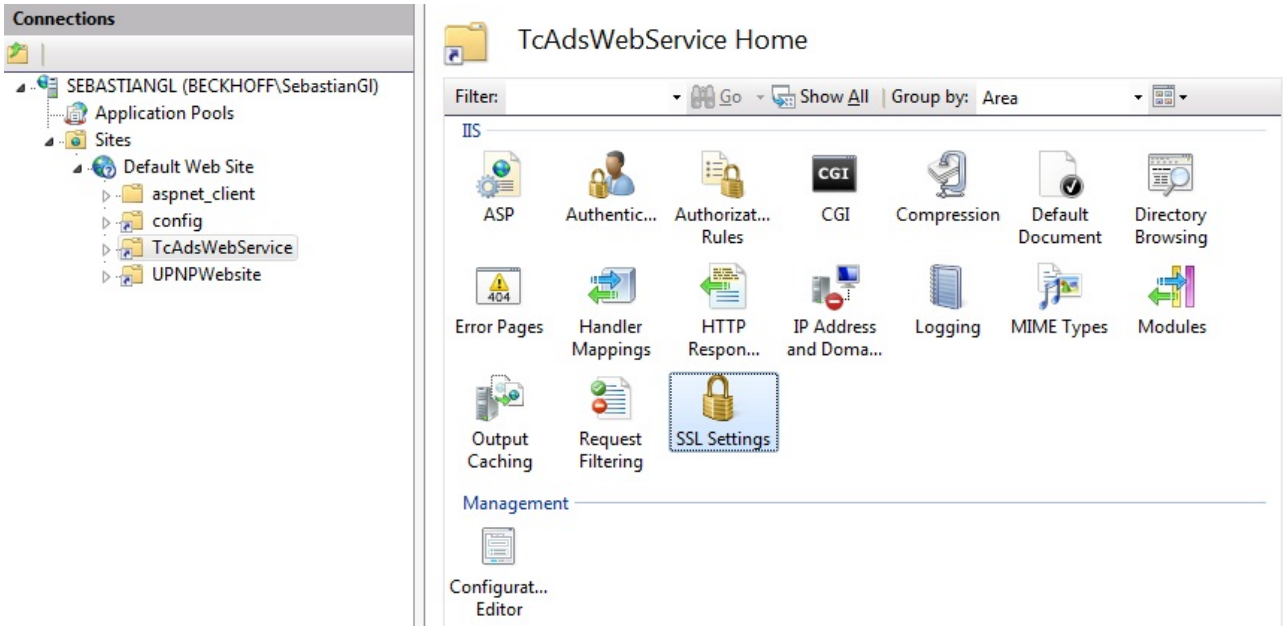


A binding for the https protocol is now available in the "Site Bindings" dialog.

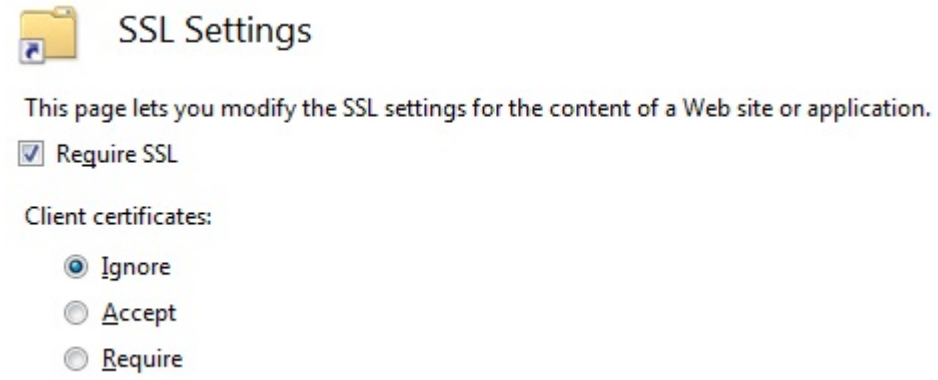


Now you can connect to all virtual directories of your website over the http and the https protocol.
Follow the next steps if you want to allow connection over the https protocol only.

Navigate to the TcAdsWebService virutal directory and open the "SSL/TLS Settings" window.



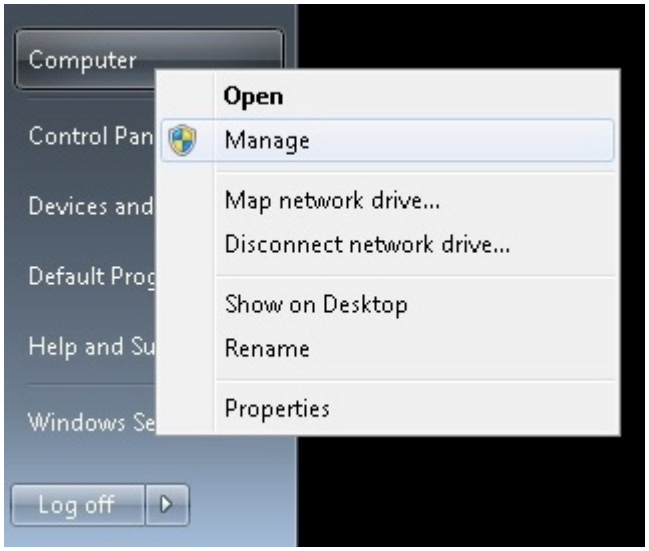
In the "SSL Settings" dialog check the "Require SSL" checkbox.



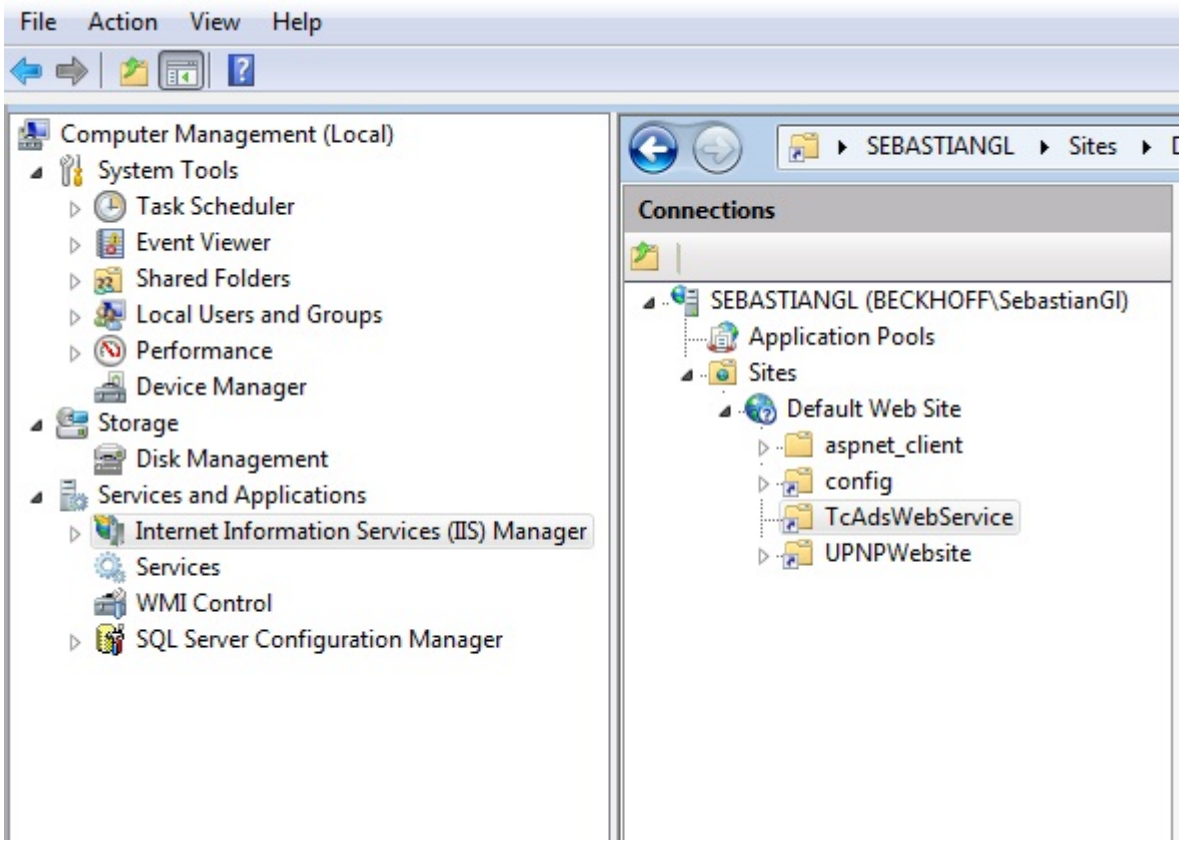
The Internet Information Services will now only allow connections over the https protocol.

NTLM Authentication

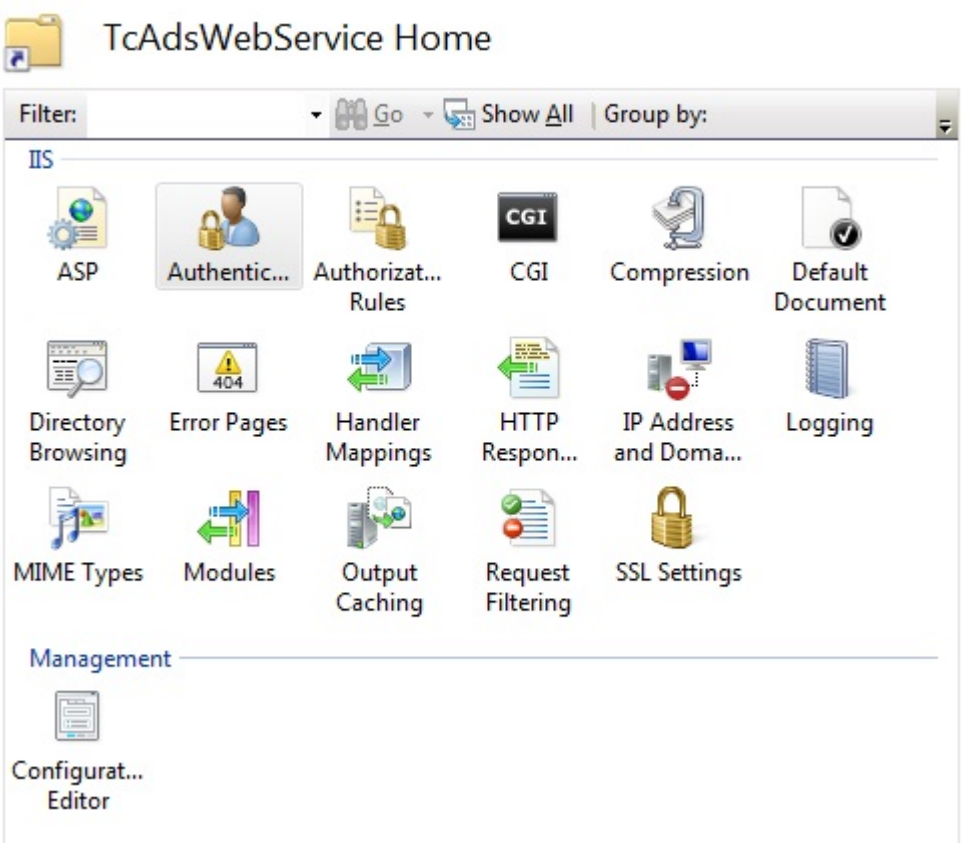
Rightclick on the "Computer" button in the Windows 7 Startmenu und press "Manage" to open the Computer Management Window.



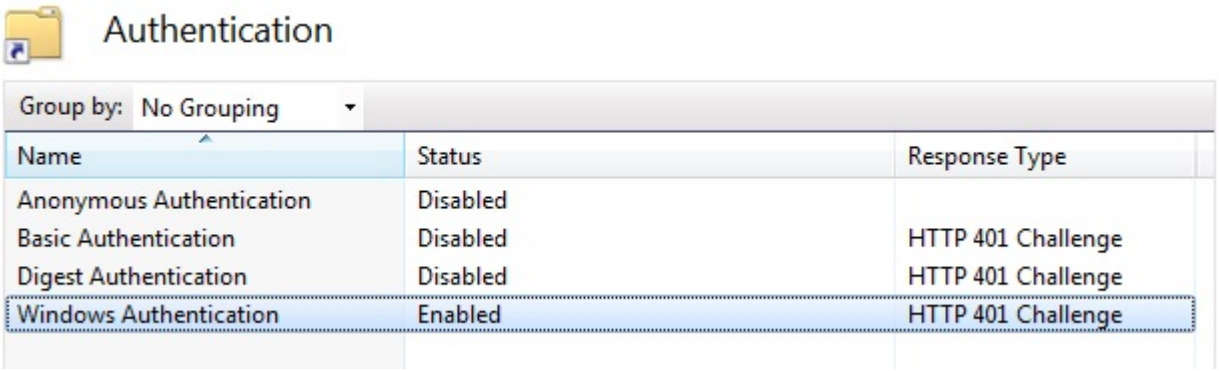
Choose the virual directory node of the TcAdsWebService in the Internet Information Services Manageer navigation.



Open the Authentication dialog for the TcAdsWebService virual directory.



Enable "Windows Authentication" and disable all other authentication methods.



NTLM is now active and required for the TcAdsWebService.

4.2 Configuration of SSL/TLS and NTLM Authentification for the TwinCAT ADS WebServices on Windows CE

SSL/TLS Configuration

In the registry create a subkey named **SSL/TLS** below **/HKEY_LOCAL_MACHINE/Comm/HTTPD/** registry key.

Create the following values in **/HKEY_LOCAL_MACHINE/Comm/HTTPD/SSL**

- **IsEnabled**
  - Type: DWORD
  - Value: 0x00000001
- **CertificateSubject**
  - Type: STRING

- Value: <Subject value of the certificate in certificate store which should be used for SSL/TLS>

Restart the HTTPD WebServer with the following console command: **services refresh htp0:**

Generating a self signed certificate with SSLCert.exe [▶ 30].

The tool is provided with the ADS API under **\TWINCATDIR\ADS Api\TcAdsWebService\SSLCert**.

NTLM Authentication

- The value **NTLM** in registry key **/HKEY_LOCAL_MACHINE/Comm/HTTPD/** has to be set to 0x00000001 to enable NTLM Authentication for the HTTPD WebServer.
- In the **/HKEY_LOCAL_MACHINE/Comm/HTTPD/VROOTS/** subkey for your website, or in the subkey named **/** if you have placed your website in **\HARD DISK\www** the value **a** has to be set to 0x00000001.
- Restart the HTTPD WebServer with the following console command: **services refresh htp0:**

4.3 SSLCert.exe

SSLCert is a console application which can be used to create self-signed certificates for SSL encoding.

Usage: SSLCert.exe DOMAINNAME

Parameters

DOMAINNAME

The name of the domain which should be secured by SSL.

Example 1

If a web resource should be reachable over the url **http://www.beckhoff.com/webui/** you have to use the following command to create a certificate with SSLCert:

```
SSLCert.exe www.beckhoff.com
```

This will generate a certificate with the common name www.beckhoff.com in the certificate store which can be used for SSL encoding. And a **www.beckhoff.com.cer** file in the working directory of the SSLCert application which can be exported on a client system as trusted root authority.

Anyway some browsers may not trust such a certificate because its self signed.

Example 2

If a web resource should be reachable over the url **http://beckhoff.com/webui/** you have to use the following command to create a certificate with SSLCert:

```
SSLCert.exe beckhoff.com
```

This will generate a certificate with the common name beckhoff.com in the certificate store which can be used for SSL encoding. And a **beckhoff.com.cer** file in the working directory of the SSLCert application which can be exported on a client system as trusted root authority.

Anyway some browsers may not trust such a certificate because its self signed.

Example 3

If a web resource should be reachable over the url **http://192.168.1.1/webui/** you have to use the following command to create a certificate with SSLCert:

```
SSLCert.exe 192.168.1.1
```

This will generate a certificate with the common name 192.168.1.1 in the certificate store which can be used for SSL encoding. And a **192.168.1.1.cer** file in the working directory of the SSLCert application which can be exported on a client system as trusted root authority.

Anyway some browsers may not trust such a certificate because its self signed.

Important:

SSLCert.exe has to be used with the "Administrator" account.
Another account does not have the required privileges to add a certificate to the system certificate store.
Even if its part of the "Administrators" group.

5 TcAdsWebService.js

The TcAdsWebService.js JavaScript Library provides objects for an easier usage of the TcAdsWebService.

Objects

The TcAdsWebService object is used as a pseudo namespace and provides access to all useable JavaScript objects.

Name	Description
AdsConnection [▶ 32]	The TcAdsWebService.AdsConnection object handles the asynchronous or synchronous AJAX communication with the TcAdsWebService by wrapping the TcAdsWebService.Client object
Client [▶ 38]	The TcAdsWebService.Client object handles the asynchronous or synchronous AJAX communication with the TcAdsWebService.
DataReader [▶ 43]	The TcAdsWebService.DataReader object can be used to read data from a Base64 encoded binary data string recieved from the TcAdsWebService.
DataWriter [▶ 47]	The TcAdsWebService.DataWriter object can be used to create a Base64 encoded binary data string for the TcAdsWebService.
TcAdsReservedIndexGroups [▶ 53]	Pseudo enumeration object.
TcAdsWebServiceDataTypes [▶ 53]	Pseudo enumeration object.
AdsState [▶ 52]	Pseudo enumeration object.
Response [▶ 50]	Provides information about the request progress and how it has finished.
RequestError [▶ 51]	Provides information about AJAX errors.
Error [▶ 51]	Provides information about ads errors.

Properties

Name	Description
nActiveReqCount	Returns a counter of active ajax requests against the TcAdsWebService.

5.1 AdsConnection

The TcAdsWebService.AdsConnection object handles the asynchronous or synchronous AJAX communication with the TcAdsWebService by wrapping the TcAdsWebService.Client object.

```
TcAdsWebService.AdsConnection(
   netid,
   port,
   service_url [optional],
   service_user [optional],
   service_password [optional]
 );
```

Parameters

netid	Type: String NetID of the target AMS Router.
port	Type: Number Port of the target AMS Router.

service_url [optional]	Type: String The url of the TcAdsWebService.
service_user [optional]	Type: String The username which should be used to access the TcAdsWebService. Set to undefined or null for anonymous access or access with session credentials.
service_password [optional]	Type: String The password which should be used to access the TcAdsWebService. Set to undefined or null for anonymous access or access with session credentials.

Functions

Name	Description
readwrite [▶ 34]	Writes data to an ADS device and then reads data from this device in a synchronous way.
readwriteAsync [▶ 34]	Writes data to an ADS device and then reads data from this device in an asynchronous way.
readState [▶ 35]	Reads the ADS status and the device status from an ADS server in a synchronous way.
readStateAsync [▶ 35]	Reads the ADS status and the device status from an ADS server in an asynchronous way.
writeControl [▶ 35]	Changes the ADS status and the device status of an ADS server in a synchronous way.
writeControlAsync [▶ 36]	Changes the ADS status and the device status of an ADS server in an asynchronous way.
write [▶ 36]	Writes data to an ADS device in a synchronous way.
writeAsync [▶ 37]	Writes data to an ADS device in an asynchronous way.
read [▶ 37]	Reads data from an ADS device in a synchronous way.
readAsync [▶ 38]	Reads data from an ADS device in an asynchronous way.
getTypeString [▶ 38]	Returns an object identifier string

Properties

Name	Description
service_url	The url of the TcAdsWebService. Dont change its value. This property is used in the underlying TcAdsWebService.Client object. You have to create a new TcAdsWebService.AdsConnection object instance to change this property.
service_user	The username which should be used to access the TcAdsWebService. Dont change its value. This property is used in the underlying TcAdsWebService.Client object. You have to create a new TcAdsWebService.AdsConnection object instance to change this property.
service_password	The password which should be used to access the TcAdsWebService. Dont change its value. This property is used in the underlying TcAdsWebService.Client object. You have to create a new TcAdsWebService.AdsConnection object instance to change this property.
timeout	The timeout for ajax request against the TcAdsWebService. Default Value: 2000
timeoutCallback	A pointer to a javascript function which is called on a timeout of an ajax request. Default Value: undefined
client	The underlying TcAdsWebService.Client object. Dont change its value. Create a new TcAdsWebService.AdsConnection object to change its value.
netid	The NetID of the target AMS Router.
port	The port of the target AMS Router.

5.1.1 readwrite

Writes data to an ADS device and then reads data from this device.

```
readwrite(
   nIndexGroup,
   nIndexOffset,
   cbRdLen,
   pwrData,
   userState,
 );
```

Parameters

nIndexGroup	Type: Number The indexGroup
nIndexOffset	Type: Number The indexOffset
cbRdLen	Type: Number Length of data to read.
pwrData	Type: Base64 encoded binary string The binary data to write.
userState	Type: Every User data which will be passed through.

Returns

This function will return a TcAdsWebService.Response object.
If successfull, the TcAdsWebService.Response.reader property contains the data in the order in which the data was requested.

5.1.2 readwriteAsync

Writes data to an ADS device and then reads data from this device.

```
readwrite(
   nIndexGroup,
   nIndexOffset,
   cbRdLen,
   pwrData,
   pCallback,
   userState,
 );
```

Parameters

nIndexGroup	Type: Number The indexGroup
nIndexOffset	Type: Number The indexOffset
cbRdLen	Type: Number Length of data to read.
pwrData	Type: Base64 encoded binary string The binary data to write.
pCallback	Type: Function pointer The callback function for asynchronous AJAX requests. Function pointer signature: *function(TcAdsWebService.Response, userState)* If successfull, the TcAdsWebService.Response.reader property contains the data in the order in which the data was requested.
userState	Type: Every User data which will be passed through.

Returns

This function will return nothing. An TcAdsWebService.Response object is returned by calling the callback function which was set by use of the pCallback parameter.
If successfull, the TcAdsWebService.Response.reader property contains the data in the order in which the data was requested.

5.1.3 readState

Reads the ADS status and the device status from an ADS server.

```
readState(
   userState,
 );
```

Parameters

userState	Type: Every User data which will be passed through.

Returns

This function will return a TcAdsWebService.Response object.
If successfull, the TcAdsWebService.DataReader object of the TcAdsWebService.Response.reader property contains the state values as 2 Byte WORD values in the order AdsState, DeviceState.

5.1.4 readStateAsync

Reads the ADS status and the device status from an ADS server.

```
readState(
   pCallback,
   userState,
 );
```

Parameters

pCallback	Type: Function pointer The callback function for asynchronous AJAX requests. Function pointer signature: *function(TcAdsWebService.Response, userState)* If successfull, the TcAdsWebService.DataReader object of the TcAdsWebService.Response.reader property contains the state values as 2 Byte WORD values in the order AdsState, DeviceState.
userState	Type: Every User data which will be passed through.

Returns

This function will return nothing. An TcAdsWebService.Response object is returned by calling the callback function which was set by use of the pCallback parameter.
If successfull, the TcAdsWebService.DataReader object of the TcAdsWebService.Response.reader property contains the state values as 2 Byte WORD values in the order AdsState, DeviceState.

5.1.5 writeControl

Changes the ADS status and the device status of an ADS server.

```
writeControl(
   adsState,
   deviceState,
   pData,
   userState,
 );
```

Parameters

adsState	Type: Number or TcAdsWebService.AdsState New ads state.
deviceState	Type: Number or TcAdsWebService.AdsState New device state.
pData	Type: Base64 encoded binary string The binary data to write.
userState	Type: Every User data which will be passed through.

Returns

This function will return a TcAdsWebService.Response object.

5.1.6 writeControlAsync

Changes the ADS status and the device status of an ADS server.

```
writeControl(
   adsState,
   deviceState,
   pData,
   pCallback,
   userState,
 );
```

Parameters

adsState	Type: Number or TcAdsWebService.AdsState New ads state.
deviceState	Type: Number or TcAdsWebService.AdsState New device state.
pData	Type: Base64 encoded binary string The binary data to write.
pCallback	Type: Function pointer The callback function for asynchronous AJAX requests. Function pointer signature: *function(TcAdsWebService.Response, userState)* If successfull, the TcAdsWebService.Response.reader property will be undefined.
userState	Type: Every User data which will be passed through.

Returns

This function will return nothing. An TcAdsWebService.Response object is returned by calling the callback function which was set by use of the pCallback parameter.

5.1.7 write

Writes data to an ADS device.

```
write(
   nIndexGroup,
   nIndexOffset,
   pData,
   userState,
 );
```

Parameters

nIndexGroup	Type: Number The indexGroup

nIndexOffset	Type: Number The indexOffset
pData	Type: Base64 encoded binary string The binary data to write.
userState	Type: Every User data which will be passed through.

Returns

This function will return a TcAdsWebService.Response object.

5.1.8 writeAsync

Writes data to an ADS device.

```
write(
    nIndexGroup,
    nIndexOffset,
    pData,
    pCallback,
    userState,
  );
```

Parameters

nIndexGroup	Type: Number The indexGroup
nIndexOffset	Type: Number The indexOffset
pData	Type: Base64 encoded binary string The binary data to write.
pCallback	Type: Function pointer The callback function for asynchronous AJAX requests. Function pointer signature: *function(TcAdsWebService.Response, userState)* If successfull, the TcAdsWebService.Response.reader property will be undefined.
userState	Type: Every User data which will be passed through.

Returns

This function will return nothing. An TcAdsWebService.Response object is returned by calling the callback function which was set by use of the pCallback parameter.

5.1.9 read

Reads data from an ADS device.

```
read(
    nIndexGroup,
    nIndexOffset,
    cbLen,
    userState
  );
```

Parameters

nIndexGroup	Type: Number The indexGroup
nIndexOffset	Type: Number The indexOffset
cbLen	Type: Number The length of read data.

userState	Type: Every User data which will be passed through.

Returns

This function will return a TcAdsWebService.Response object.
If successfull, the TcAdsWebService.Response.reader property contains the data in the order in which the data was requested.

5.1.10 readAsync

Reads data from an ADS device.

```
readAsync(
    nIndexGroup,
    nIndexOffset,
    cbLen,
    pCallback,
    userState
  );
```

Parameters

nIndexGroup	Type: Number The indexGroup
nIndexOffset	Type: Number The indexOffset
cbLen	Type: Number The length of read data.
pCallback	Type: Function pointer The callback function for asynchronous AJAX requests. Function pointer signature: *function(TcAdsWebService.Response, userState)* If successfull, the TcAdsWebService.Response.reader property contains the data in the order in which the data was requested.
userState	Type: Every User data which will be passed through.

Returns

This function will return nothing. An TcAdsWebService.Response object is returned by calling the callback function which was set by use of the pCallback parameter.
If successfull, the TcAdsWebService.Response.reader property contains the data in the order in which the data was requested.

5.1.11 getTypeString

Returns an object identifier string.

```
getTypeString( );
```

Returns

Returns an object identifier string.
Expected value: TcAdsWebservice.AdsConnection

5.2 Client

The TcAdsWebService.Client object handles the asynchronous or synchronous AJAX communication with the TcAdsWebService.

```
TcAdsWebService.Client(
   sServiceUrl,
   sServiceUser,
   sServicePassword
 );
```

Parameters

sServiceUrl	Type: String The url of the TcAdsWebService.
sServiceUser	Type: String The username which should be used to access the TcAdsWebService. Set to undefined or null for anonymous access or access with session credentials.
sServicePassword	Type: String The password which should be used to access the TcAdsWebService. Set to undefined or null for anonymous access or access with session credentials.

Functions

Name	Description
readwrite [▶ 39]	Writes data to an ADS device and then reads data from this device.
readState [▶ 40]	Reads the ADS status and the device status from an ADS server.
writeControl [▶ 41]	Changes the ADS status and the device status of an ADS server.
write [▶ 42]	Writes data to an ADS device.
read [▶ 42]	Reads data from an ADS device.
getTypeString [▶ 43]	Returns an object identifier string.

5.2.1 readwrite

Writes data to an ADS device and then reads data from this device.

```
readwrite(
   sNetId,
   nPort,
   nIndexGroup,
   nIndexOffset,
   cbRdLen,
   pwrData,
   pCallback,
   userState,
   ajaxTimeout,
   ajaxTimeoutCallback,
   async
 );
```

Parameters

sNetId	Type: String NetID of the target AMS Router.
nPort	Type: Number Port of the target AMS Router.
nIndexGroup	Type: Number The indexGroup
nIndexOffset	Type: Number The indexOffset
cbRdLen	Type: Number Length of data to read.
pwrData	Type: Base64 encoded binary string The binary data to write.

pCallback	Type: Function pointer The callback function for asynchronous AJAX requests. Function pointer signature: *function(TcAdsWebService.Response, userState)* If successfull, the TcAdsWebService.Response.reader property contains the data in the order in which the data was requested.
userState	Type: Every User data which will be passed through.
ajaxTimeout	Type: Number The timeout value for AJAX requests in milliseconds.
ajaxTimeoutCallback	Type: Function pointer The callback function for AJAX timeouts. Function pointer signature: *function()*
async	Type: Bool Determines whether asynchronous request should be used or not.

Returns

If the async parameter is set to false, this function will return a TcAdsWebService.Response object. Otherwise it returns nothing.

If successfull, the TcAdsWebService.Response.reader property contains the data in the order in which the data was requested.

5.2.2 readState

Reads the ADS status and the device status from an ADS server.

```
readState(
   sNetId,
   nPort,
   pCallback,
   userState,
   ajaxTimeout,
   ajaxTimeoutCallback,
   async
 );
```

Parameters

sNetId	Type: String NetID of the target AMS Router.
nPort	Type: Number Port of the target AMS Router.
pCallback	Type: Function pointer The callback function for asynchronous AJAX requests. Function pointer signature: *function(TcAdsWebService.Response, userState)* If successfull, the TcAdsWebService.DataReader object of the TcAdsWebService.Response.reader property contains the state values as 2 Byte WORD values in the order AdsState, DeviceState.
userState	Type: Every User data which will be passed through.
ajaxTimeout	Type: Number The timeout value for AJAX requests in milliseconds.
ajaxTimeoutCallback	Type: Function pointer The callback function for AJAX timeouts. Function pointer signature: *function()*
async	Type: Bool Determines whether asynchronous request should be used or not.

Returns

If the async parameter is set to false, this function will return a TcAdsWebService.Response object. Otherwise it returns nothing.

If successfull, the TcAdsWebService.DataReader object of the TcAdsWebService.Response.reader property contains the state values as 2 Byte WORD values in the order AdsState, DeviceState.

5.2.3 writeControl

Changes the ADS status and the device status of an ADS server.

```
writeControl(
   sNetId,
   nPort,
   adsState,
   deviceState,
   pData,
   pCallback,
   userState,
   ajaxTimeout,
   ajaxTimeoutCallback,
   async
 );
```

Parameters

sNetId	Type: String NetID of the target AMS Router.
nPort	Type: Number Port of the target AMS Router.
adsState	Type: Number or TcAdsWebService.AdsState
New ads state.	
deviceState	Type: Number or TcAdsWebService.AdsState
New device state.	
pData	Type: Base64 encoded binary string The binary data to write.
pCallback	Type: Function pointer The callback function for asynchronous AJAX requests. Function pointer signature: *function(TcAdsWebService.Response, userState)* If successfull, the TcAdsWebService.Response.reader property will be undefined.
userState	Type: Every User data which will be passed through.
ajaxTimeout	Type: Number The timeout value for AJAX requests in milliseconds.
ajaxTimeoutCallback	Type: Function pointer The callback function for AJAX timeouts. Function pointer signature: *function()*
async	Type: Bool Determines whether asynchronous request should be used or not.

Returns

If the async parameter is set to false, this function will return a TcAdsWebService.Response object. Otherwise it returns nothing.

If successfull, the TcAdsWebService.Response.reader property will be undefined.

5.2.4 write

Writes data to an ADS device.

```
write(
    sNetId,
    nPort,
    nIndexGroup,
    nIndexOffset,
    pData,
    pCallback,
    userState,
    ajaxTimeout,
    ajaxTimeoutCallback,
    async
  );
```

Parameters

sNetId	Type: String NetID of the target AMS Router.
nPort	Type: Number Port of the target AMS Router.
nIndexGroup	Type: Number The indexGroup
nIndexOffset	Type: Number The indexOffset
pData	Type: Base64 encoded binary string The binary data to write.
pCallback	Type: Function pointer The callback function for asynchronous AJAX requests. Function pointer signature: *function(TcAdsWebService.Response, userState)* If successfull, the TcAdsWebService.Response.reader property will be undefined.
userState	Type: Every User data which will be passed through.
ajaxTimeout	Type: Number The timeout value for AJAX requests in milliseconds.
ajaxTimeoutCallback	Type: Function pointer The callback function for AJAX timeouts. Function pointer signature: *function()*
async	Type: Bool Determines whether asynchronous request should be used or not.

Returns

If the async parameter is set to false, this function will return a TcAdsWebService.Response object. Otherwise it returns nothing.

If successfull, the TcAdsWebService.Response.reader property will be undefined.

5.2.5 read

Reads data from an ADS device.

```
read(
    sNetId,
    nPort,
    nIndexGroup,
    nIndexOffset,
    cbLen,
    pCallback,
    userState,
    ajaxTimeout,
```

```
    ajaxTimeoutCallback,
    async
  );
```

Parameters

sNetId	Type: String NetID of the target AMS Router.
nPort	Type: Number Port of the target AMS Router.
nIndexGroup	Type: Number The indexGroup
nIndexOffset	Type: Number The indexOffset
cbLen	Type: Number The length of read data.
pCallback	Type: Function pointer The callback function for asynchronous AJAX requests. Function pointer signature: *function(TcAdsWebService.Response, userState)* If successfull, the TcAdsWebService.Response.reader property contains the data in the order in which the data was requested.
userState	Type: Every User data which will be passed through.
ajaxTimeout	Type: Number The timeout value for AJAX requests in milliseconds.
ajaxTimeoutCallback	Type: Function pointer The callback function for AJAX timeouts. Function pointer signature: *function()*
async	Type: Bool Determines whether asynchronous request should be used or not.

Returns

If the async parameter is set to false, this function will return a TcAdsWebService.Response object. Otherwise it returns nothing.

If successfull, the TcAdsWebService.Response.reader property contains the data in the order in which the data was requested.

5.2.6 getTypeString

Returns an object identifier string.

```
getTypeString( );
```

Returns

Returns an object identifier string.
Expected value: TcAdsWebservice.Client

5.3 DataReader

The TcAdsWebService.DataReader object can be used to read data from a Base64 encoded binary data string recieved from the TcAdsWebService.

```
TcAdsWebService.DataReader(
    data
);
```

Parameters

data	Type: String Base64 encoded binary data.

Table 1: Properties

Name	Description
offset	Actual position in data buffer.
decodedData	Binary string data buffer.

Functions

Name	Description
readSINT [▶ 44]	Reads 1 byte from the decodedData property and returns it as signed 1 byte numeric value.
readINT [▶ 44]	Reads 2 byte from the decodedData property and returns it as signed 2 byte numeric value.
readDINT [▶ 45]	Reads 4 byte from the decodedData property and returns it as signed 4 byte numeric value.
readBYTE [▶ 45]	Reads 1 byte from the decodedData property and returns it as unsigned 1 byte numeric value.
readWORD [▶ 45]	Reads 2 bytes from the decodedData property and returns it as unsigned 2 byte numeric value.
readDWORD [▶ 45]	Reads 4 bytes from the decodedData property and returns it as unsigned 4 byte numeric value.
readBOOL [▶ 45]	Reads 1 byte from the decodedData property and returns it as boolean value.
readString [▶ 45]	Reads a string from the decodedData property.
readREAL [▶ 46]	Reads 4 byte from the decodedData property and returns it as 4 byte floating point number value.
readLREAL [▶ 46]	Reads 8 byte from the decodedData property and returns it as 8 byte floating point number
getTypeString [▶ 46]	Returns an object identifier string.

5.3.1 readSINT

Reads 1 byte from the decodedData property and returns it as signed 1 byte numeric value.

```
readSINT();
```

Returns

Signed 1 byte numeric value.

5.3.2 readINT

Reads 2 byte from the decodedData property and returns it as signed 2 byte numeric value.

```
readINT();
```

Returns

Signed 2 byte numeric value.

5.3.3 readDINT

Reads 4 byte from the decodedData property and returns it as signed 4 byte numeric value.

```
readDINT();
```

Returns

Signed 4 byte numeric value.

5.3.4 readBYTE

Reads 1 byte from the decodedData property and returns it as unsigned 1 byte numeric value.

```
readBYTE();
```

Returns

Unsigned 1 byte numeric value.

5.3.5 readWORD

Reads 2 bytes from the decodedData property and returns it as unsigned 2 byte numeric value.

```
readWORD();
```

Returns

Unsigned 2 byte numeric value.

5.3.6 readDWORD

Reads 4 bytes from the decodedData property and returns it as unsigned 4 byte numeric value.

```
readDWORD();
```

Returns

Unsigned 4 byte numeric value.

5.3.7 readBOOL

Reads 1 byte from the decodedData property and returns it as boolean value.

```
readBOOL();
```

Returns

Boolean value.

5.3.8 readString

Reads a string from the decodedData property.

```
readString(
    length
);
```

Parameters

length	The length of the string to read.

Returns

String value.

5.3.9 readStringZero

Reads a string from the decodedData property and places an zero termination for the first zero byte in the binary string.

```
readStringZero(
    length
);
```

Parameters

length	The length of the string to read.

Returns

String value.

5.3.10 readREAL

Reads 4 byte from the decodedData property and returns it as 4 byte floating point number value.

```
readREAL();
```

Returns

4 byte floating point number.

5.3.11 readLREAL

Reads 8 byte from the decodedData property and returns it as 8 byte floating point number.

```
readLREAL();
```

Returns

8 byte floating point number.

5.3.12 getByteLength

Reads a string from the decodedData property and places an zero termination for the first zero byte in the binary string.

```
getByteLength( );
```

Parameters

-

Returns

Returns the length of the underlying byte string as number value.

5.3.13 getTypeString

Returns an object identifier string.

```
TcAdsWebService.DataReader.getTypeString();
```

Returns

Returns an object identifier string.
Expected value: TcAdsWebservice.DataReader

5.4 DataWriter

The TcAdsWebService.DataWriter object can be used to create a Base64 encoded binary data string for the TcAdsWebService.

```
TcAdsWebService.DataWriter( );
```

Functions

Name	Description
getBase64EncodedData [▶ 47]	Returns the internal byte buffer as Base64 encoded binary string.
writeSINT [▶ 47]	Writes a 1 byte signed numeric value to the internal byte buffer.
writeINT [▶ 47]	Writes a 2 byte signed numeric value to the internal byte buffer.
writeDINT [▶ 48]	Writes a 4 byte signed numeric value to the internal byte buffer.
writeBYTE [▶ 48]	Writes a 1 byte unsigned numeric value to the internal byte buffer.
writeWORD [▶ 48]	Writes a 2 byte unsigned numeric value to the internal byte buffer.
writeDWORD [▶ 48]	Writes a 4 byte unsigned numeric value to the internal byte buffer.
writeBOOL [▶ 49]	Writes a 1 byte boolean value to the internal byte buffer.
writeString [▶ 49]	Writes a string of given length to the internal byte buffer.
writeREAL [▶ 49]	Writes a 4 byte floating point number to the internal byte buffer.
writeLREAL [▶ 49]	Writes a 8 byte floating point number to the internal byte buffer.
getTypeString [▶ 49]	Returns an object identifier string.

5.4.1 getBase64EncodedData

Returns the internal byte buffer as Base64 encoded binary string.

```
getBase64EncodedData();
```

Returns

Base64 encoded binary string.

5.4.2 writeSINT

Writes a 1 byte signed numeric value to the internal byte buffer.

```
writeSINT(
    value
);
```

Parameters

value	Type: Number 1 byte signed numeric value.

5.4.3 writeINT

Writes a 2 byte signed numeric value to the internal byte buffer.

```
writeINT(
    value
);
```

Parameters

value	Type: Number 2 byte signed numeric value.

5.4.4 writeDINT

Writes a 4 byte signed numeric value to the internal byte buffer.

```
writeDINT(
    value
);
```

Parameters

value	Type: Number 4 byte signed numeric value.

5.4.5 writeBYTE

Writes a 1 byte unsigned numeric value to the internal byte buffer.

```
writeBYTE(
    value
);
```

Parameters

value	Type: Number 1 byte unsigned numeric value.

5.4.6 writeWORD

Writes a 2 byte unsigned numeric value to the internal byte buffer.

```
writeWORD(
    value
);
```

Parameters

value	Type: Number A 2 byte unsigned numeric value.

5.4.7 writeDWORD

Writes a 4 byte unsigned numeric value to the internal byte buffer.

```
writeDWORD(
    value
);
```

Parameters

value	Type: Number 4 byte unsigned numeric value.

5.4.8 writeBOOL

Writes a 1 byte boolean value to the internal byte buffer.

```
writeBOOL(
    value
);
```

Parameters

value	Type: Boolean 1 byte boolean value.

5.4.9 writeString

Writes a string of given length to the internal byte buffer.

```
writeString(
    value,
    length
);
```

Parameters

value	Type: String The string to write.
length	Type: Number The length of the string.

5.4.10 writeREAL

Writes a 4 byte floating point number to the internal byte buffer.

```
writeREAL(
    value
);
```

Parameters

value	Type: Number 4 byte floating point number.

5.4.11 writeLREAL

Writes a 8 byte floating point number to the internal byte buffer.

```
writeLREAL(
    value
);
```

Parameters

value	Type: Number 8 byte floating point number.

5.4.12 getTypeString

Returns an object identifier string.

```
getTypeString();
```

Returns

Returns an object identifier string.
Expected value: TcAdsWebservice.DataWriter

5.5 Response

Provides information about the request progress and how it has finished.

```
Response(
   hasError,
   error,
   reader,
   isBusy
 );
```

Parameters

hasError	Type: Bool Determines wheter the request has finished with an error or not.
error	Type: TcAdsWebService.Error, TcAdsWebService.RequestError Further information about an error. If no further information exists or no error occured, this value has to be null or undefined.
reader	Type: TcAdsWebSerivce.DataReader The returned data as TcAdsWebService.DataReader object. Set to null or undefined if an error occured.
isBusy	Type: Bool Determines whether the request has finished or if it is still in progress.

Propertiess

Name	Description
hasError	Type: Bool Determines wheter the request has finished with an error or not.
error	Type: TcAdsWebService.Error, TcAdsWebService.RequestError Further information about an error. If no further information exists or no error occured, this value is set to null or undefined.
reader	Type: TcAdsWebSerivce.DataReader The returned data. If an error occured this value is set to null or undefined.
isBusy	Type: Bool Determines whether the request has finished or if it is still in progress.

Functions

Name	Description
getTypeString [▶ 50]	Returns an object identifier string.

5.5.1 getTypeString

Returns an object identifier string.

```
getTypeString();
```

Returns

Returns an object identifier string.
Expected value: TcAdsWebservice.Response

5.6 Error

Provides information about ads errors.

```
Error(
    errorMessage,
    errorCode,
    innerError [optional]
);
```

Parameters

errorMessage	Type: String Ads error message.
errorCode	Type: Number Ads error code.
innerError [optional]	Type: Every The error which caused this error response. This maybe an JavaScript Exception

Properties

Name	Description
errorMessage	Ads error message Type: String
errorCode	Ads error code. Type: Number
innerError	Type: Every The error which caused this error response. This maybe an JavaScript Exception.

Functions

Name	Description
getTypeString [▶ 51]	Returns an object identifier string.

5.6.1 getTypeString

Returns an object identifier string.

```
getTypeString( );
```

Returns

Returns an object identifier string.
Expected value: TcAdsWebservice.Error

5.7 RequestError

Provides information about AJAX errors.

```
ResquestError(
    requestStatus,
    requestStatusText
);
```

Parameters

requestStatus	Type: Number The status value of the used XmlHttpRequest object.

requestStatusText	Type: String The statusText value of the used XmlHttpRequest object.

Propertiess

Name	Description
requestStatus	Type: Number The status value of the used XmlHttpRequest object.
requestStatusText	Type: String The statusText value of the used XmlHttpRequest object.

Functions

Name	Description
getTypeString [▶ 52]	Returns an object identifier string.

5.7.1 getTypeString

Returns an object identifier string.

```
getTypeString();
```

Returns

Returns an object identifier string.
Expected value: TcAdsWebservice.RequestError

5.8 AdsState

Pseudo enumeration of AdsState values.

```
AdsState = {
    "INVALID"      :   0,
    "IDLE"         :   1,
    "RESET"        :   2,
    "INIT"         :   3,
    "START"        :   4,
    "RUN"          :   5,
    "STOP"         :   6,
    "SAVECFG"      :   7,
    "LOADCFG"      :   8,
    "POWERFAILURE" :   9,
    "POWERGOOD"    :  10,
    "ERROR"        :  11,
    "SHUTDOWN"     :  12,
    "SUSPEND"      :  13,
    "RESUME"       :  14,
    "CONFIG"       :  15,
};
```

Properties

Name	Description
INVALID	Invalided state
IDLE	Idle state
RESET	Reset state
INIT	Initialized
START	Started
RUN	Running
STOP	Stopped

Name	Description
SAVECFG	Saved configuration
LOADCFG	Load configuration
POWERFAILURE	Power failure
POWERGOOD	Power good
ERROR	Error state
SHUTDOWN	Shutting down
SUSPEND	Suspended
RESUME	Resumed
CONFIG	Config mode

5.9 TcAdsWebServiceDataTypes

```
TcAdsWebServiceDataTypes = {
    "String"         : 0,
    "BOOL"           : 1,
    "Integer"        : 2,
    "UnsignedInteger": 3,
    "LREAL"          : 4,
    "REAL"           : 5
};
```

Properties

Name	Description
String	-
BOOL	-
Integer	-
UnsignedInteger	-
LREAL	-
REAL	-

5.10 TcAdsReservedIndexGroups

Pseudo enumeration of reserverd IndexGroups

```
TcAdsReservedIndexGroups = {
    "PlcRWMX"            : 16416,
    "PlcRWMB"            : 16416,
    "PlcRWRB"            : 16432,
    "PlcRWDB"            : 16448,
    "SymbolTable"        : 61440,
    "SymbolName"         : 61441,
    "SymbolValue"        : 61442,
    "SymbolHandleByName" : 61443,
    "SymbolValueByName"  : 61444,
    "SymbolValueByHandle": 61445,
    "SymbolReleaseHandle": 61446,
    "SymbolInfoByName"   : 61447,
    "SymbolVersion"      : 61448,
    "SymbolInfoByNameEx" : 61449,
    "SymbolDownload"     : 61450,
    "SymbolUpload"       : 61451,
    "SymbolUploadInfo"   : 61452,
    "SymbolNote"         : 61456,
    "IOImageRWIB"        : 61472,
    "IOImageRWIX"        : 61473,
    "IOImageRWOB"        : 61488,
    "IOImageRWOX"        : 61489,
    "IOImageClearI"      : 61504,
    "IOImageClearO"      : 61520,
    "DeviceData"         : 61696
};
```

Properties

Name	Description
PlcRWMB	-
PlcRWMX	-
PlcRWRB	-
PlcRWDB	-
SymbolTable	-
SymbolName	-
SymbolValue	-
SymbolHandleByName	-
SymbolValueByName	-
SymbolValueByHandle	-
SymbolReleaseHandle	-
SymbolInfoByName	-
SymbolVersion	-
SymbolInfoByNameEx	-
SymbolDownload	--
SymbolUpload	-
SymbolUploadInfo	-
SymbolNote	-
IOImageRWIB	-
IOImageRWIX	-
IOImageRWOB	-
IOImageRWOX	-
IOImageClearI	-
IOImageClearO	-
DeviceData	-

6 Samples

6.1 Samples: ADS Web Service

Sample	Download link
Implementation Consumer in C# [▶ 55]	https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488721931/.exe
Implementation Consumer in C++ [▶ 63]	https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488723339/.zip
Implementation Consumer in Java [▶ 71]	https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488724747/.zip
Implementation ASP.NET 2.0 AJAX "Atlas [▶ 73]"	https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488726155/.zip https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488727563/.zip https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488728971/.zip
Implementation Consumer in Delphi 8 for .NET [▶ 66]	https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488730379/.exe

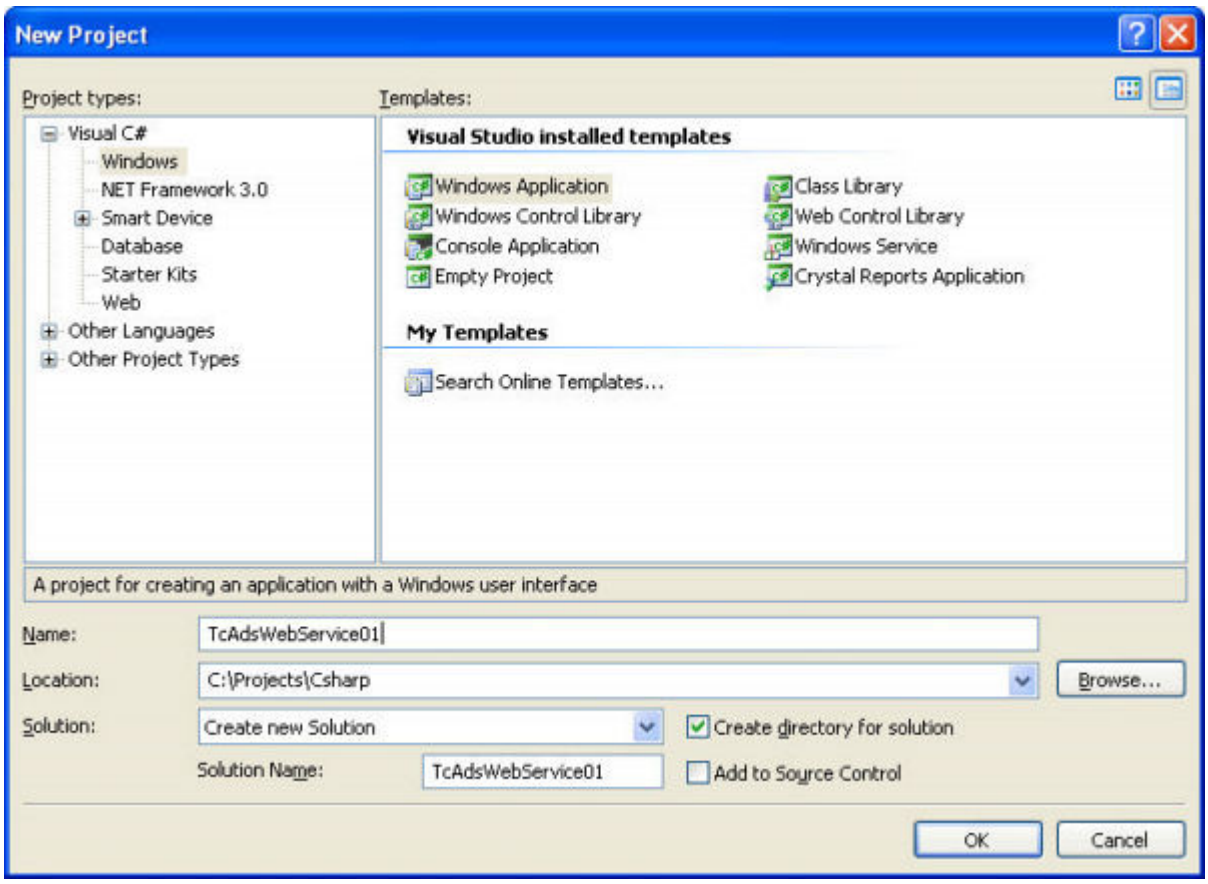
6.1.1 Consumer in C# to read and write PLC variables

This sample explains how to create an ADS-WebService-Consumer to read and write PLC-Variables via ADS-HTTP.

Creating a WebService-Consumers within the Microsoft VisualStudio.NET

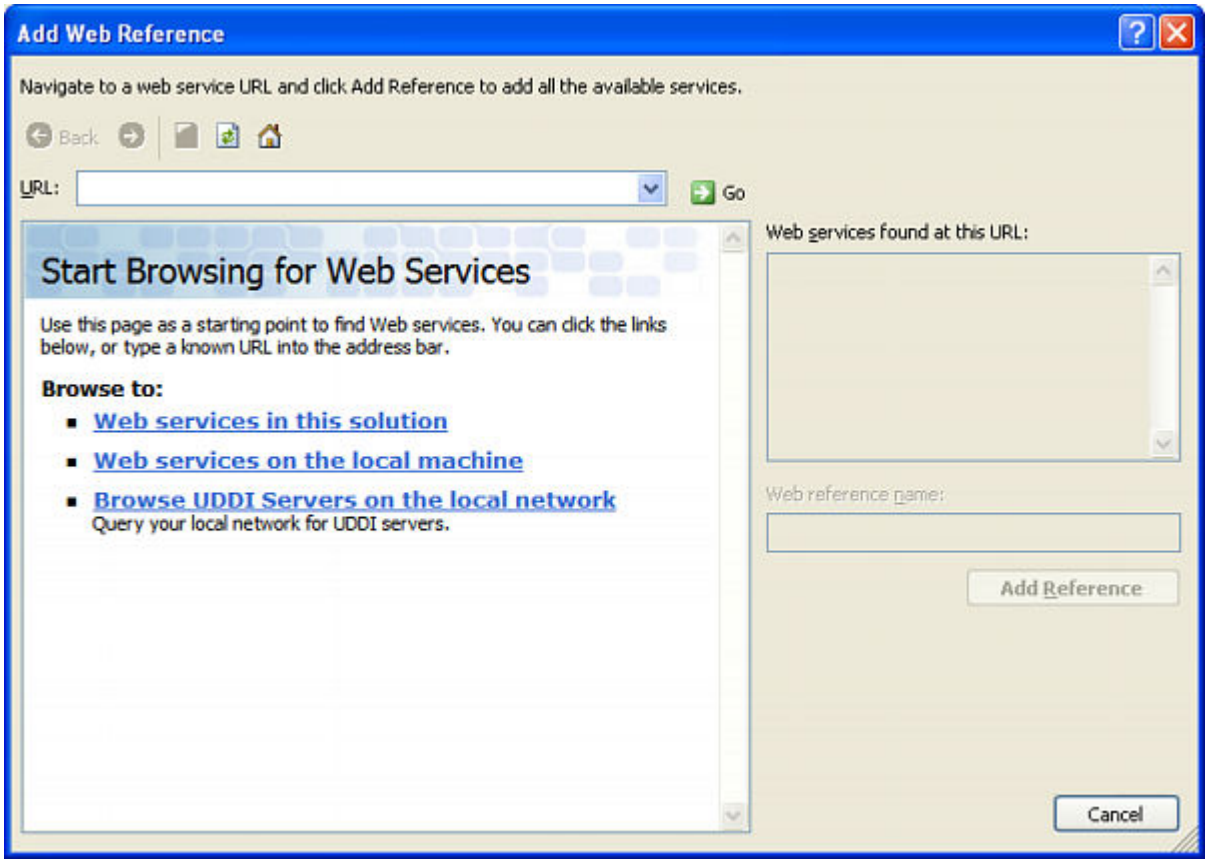
In Visual Studio select "New Project" and create a new C# - Windows Application.

- By selecting the template "Windows-Application" the new application is only running on .NET-platforms.
- By selecting the template "Smart Device Application" the new application is running on .NET-platforms and on .NET-CompactFramework (CE-) platforms.
  By selecting the .NET-CompactFramework (CE-) platform restrictions may occure.

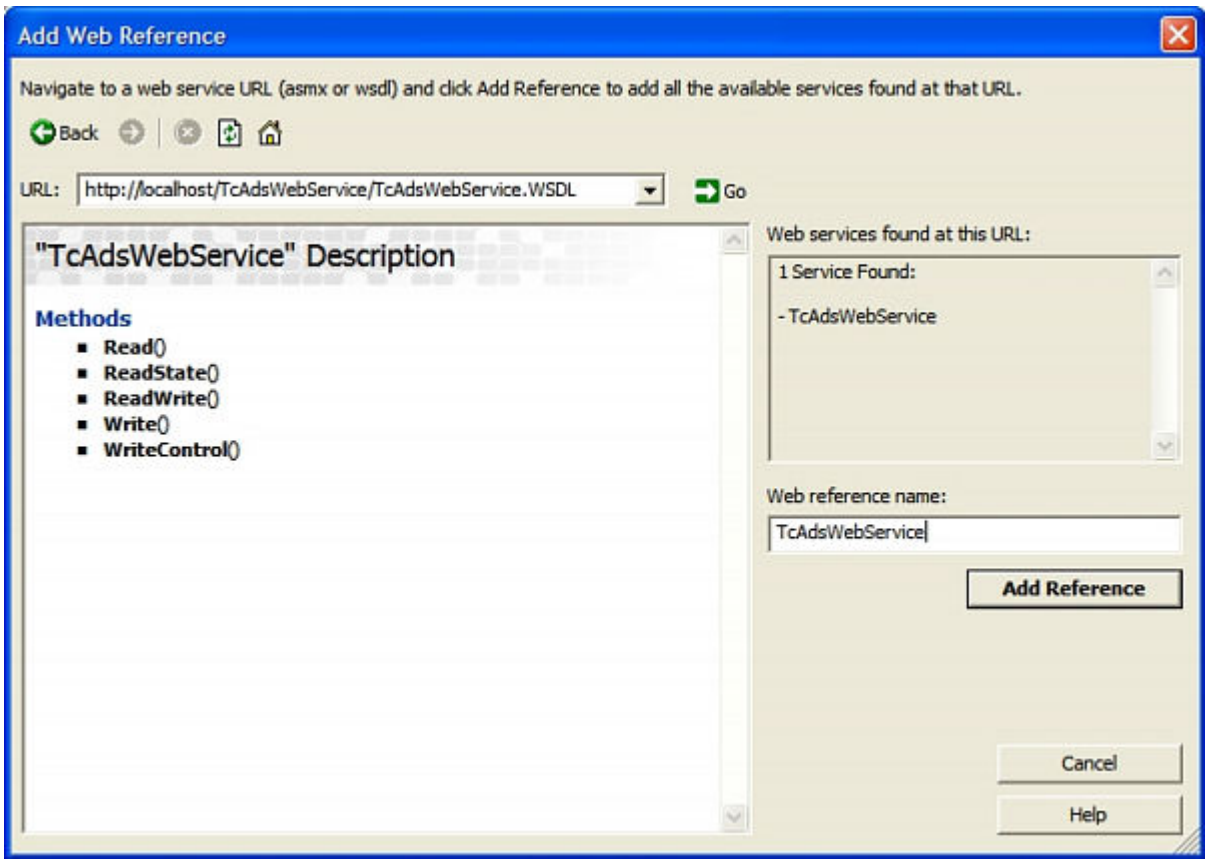


After creating the projects go to "Projects" "Add Web Reference...".

In the following dialog you may insert the URL of the WebService-WSDL-file or you can browse for it by clicking on "Web services on the local machine".



Name the reference and click the button "Add Reference" to complete .



To catch SOAP-Exceptions add the following line to the first lines of the source-code:

```
using System.Web.Services.Protocols;
```

To create a WebService-object insert the following line after "public class *NameOfYourForm*: System.Windows.Forms.Form":

```
TcAdsWebService.TcAdsWebService TcWebService = new TcAdsWebService.TcAdsWebService();
```

Add the following line in the constructor ("public *NameOfYourForm*") after "InitializeComponent();":

```
TcWebService.Url = "http://localhost/TcAdsWebService/TcAdsWebService.dll";

// Sample 1 :
TcWebService.Url = "http://192.168.0.2/TcAdsWebService/TcAdsWebService.dll";

// Sample 2 :
TcWebService.Url = "http://CX_xxxxxx/TcAdsWebService/TcAdsWebService.dll";
```

The address must be the URL of your TwinCAT ADS WebService-DLL-file.

Converting variables to byte-arrays

Variables must be converted to byte-arrays if they should be used with the ADS-WebService. Therefore it is necessary to be able to convert variables to byte-arrays and vice versa:

This could be realized with the BitConverter-class:

```
byte[] abDataBool = new byte[1];
abDataBool = BitConverter.GetBytes(bValueBool);

byte[] abDataInt = new byte[2];
abDataInt = BitConverter.GetBytes((Int16)iValueInt)
```

In this example a "bool"- and an "int"-variable were converted to byte-arrays.

To convert a byte-array back to a variable use the BitConverter-class as followed:

```
byte[] abDataBuffer = new byte[3];
bool bVarBool = BitConverter.ToBoolean(abDataBuffer,0);
int iVarInt = BitConverter.ToInt16(abDataBuffer,1);
```

Strings are converted with the the "ASCII-Encoder":

```
string szValueString;
byte[] abDataString = new byte[81];
System.Text.ASCIIEncoding encoder = new System.Text.ASCIIEncoding();
encoder.GetBytes(szValueString, 0, encoder.GetByteCount(szValueString), abDataString, 0);
```

To convert a byte-array back to a string use the following code:

```
byte[] abDataBuffer = new byte[81];
System.Text.ASCIIEncoding encoder = new System.Text.ASCIIEncoding();
string VarString = encoder.GetString(abDataBuffer, 3, 81);
```

You can get more detailed information about this methods and classes in literature for C#-Application-Developement.

Reading PLC-variables

To read a PLC-Variable use the "read"-method of the WebService-object:

```
WebService.Read(string netId, int nPort,
System.UInt32 indexGroup, System.UInt32 indexOffset, int cblen, out
System.Byte[] ppData);
```

- **string netId**: String indicating the AMS-Net-Id of the PLC
- **int nPort**: Port-number of the Runtime-System
- **System.UInt32 indexGroup**: IndexGroup of the variables to read
- **System.UInt32 indexOffset**: First byte to read
- **int cblen**: Number of bytes to read
- **out System.Byte[] ppData**: Byte-array to save the PLC-data into

Example:

```
try
{
    TcWebService.Read("192.168.0.2.1.1", 801, 0x4020, 0, 84, out abDataBuffer);
}
catch (SoapException ex)
{
    MessageBox.Show(ex.Message);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

Writing PLC-variables

To write a PLC-Variable use the "write"-method of the WebService-object:

```
WebService.Write(string netId, int nPort,
System.UInt32 indexGroup, System.UInt32 indexOffset, System.Byte[]
pData);
```

- **string netId**: String indicating the AMS-Net-Id of the PLC
- **int nPort**: Port-number of the Runtime-System
- **System.UInt32 indexGroup**: IndexGroup of the variable
- **System.UInt32 indexOffset:** First byte to write to
- **System.Byte[] pData:** Byte-array containing the data to write

Example:

```
byte[] Data = new byte[10];
try
{
    TcWebService.Write("192.168.0.2.1.1", 801, 0x4020, 0, Data);
}
catch (SoapException ex)
{
```

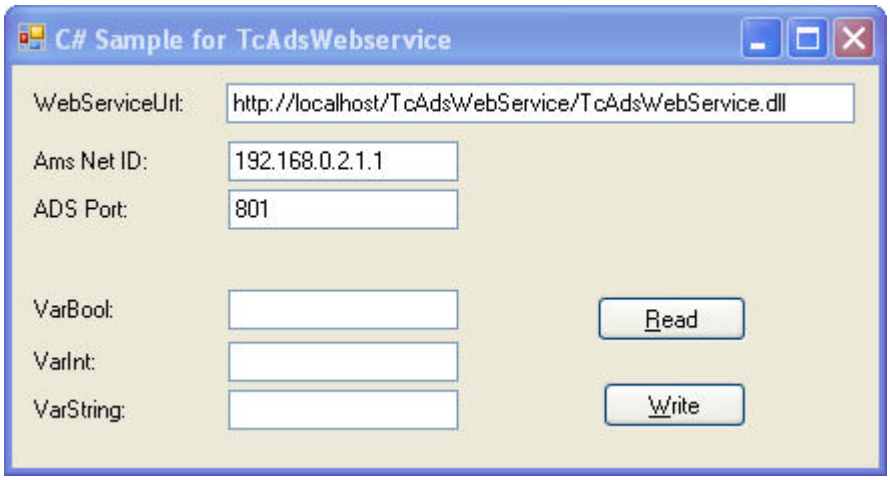
```
    MessageBox.Show(ex.Message);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

Sample-Application:

PLC-program, IEC1131 declaration of the variables

```
PROGRAM MAIN
VAR
    PlcVarBool    AT %MX0.0  : BOOL := TRUE;
    PlcVarInt     AT %MW1    : INT := 1234;
    PlcVarString  AT %MD3    : STRING := 'Hello Automation';
END_VAR
```

C#-Application, the layout :



The source-code:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Web.Services.Protocols;    // Catches the SoapExceptions

namespace WebServiceTestConsumer
{
    public class frmWebServiceTestConsumer : System.Windows.Forms.Form
    {
        private System.ComponentModel.Container components = null;

        #region forms
        private System.Windows.Forms.Label lblBool;
        private System.Windows.Forms.Label lblInt;
        private System.Windows.Forms.Label lblString;
        private System.Windows.Forms.TextBox txtBool;
        private System.Windows.Forms.TextBox txtInt;
        private System.Windows.Forms.TextBox txtString;
        private System.Windows.Forms.Button btnRead;
        private System.Windows.Forms.Button btnWrite;
        #endregion

        #region variables and instances

        // Encodes byte-array to Strings and vice versa
        private System.Text.ASCIIEncoding           encoder         =       new     System.Text.ASCI
IEncoding();

        // Instance of the TcAdsWebService
        private TcAdsWebService.TcAdsWebService TcWebService    =       new     TcAdsWebService.TcAd
sWebService();

        private string  szWebServiceUrl     = "http://192.168.0.2/TcAdsWebService/
```

```
TcAdsWebService.dll";
        public  string  szAmsNetId              =       "192.168.0.2.1.1";
        public  int         iPort           =       801;
        public  UInt32  iIndexGroup             =       0x4020;

        #endregion


        public frmWebServiceTestConsumer()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            // Links the WebService with its library
            TcWebService.Url = szWebServiceUrl;
        }

        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }


        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            this.lblBool = new System.Windows.Forms.Label();
            this.lblInt = new System.Windows.Forms.Label();
            this.lblString = new System.Windows.Forms.Label();
            this.txtBool = new System.Windows.Forms.TextBox();
            this.txtInt = new System.Windows.Forms.TextBox();
            this.txtString = new System.Windows.Forms.TextBox();
            this.btnRead = new System.Windows.Forms.Button();
            this.btnWrite = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // lblBool
            //
            this.lblBool.Location = new System.Drawing.Point(40, 24);
            this.lblBool.Name = "lblBool";
            this.lblBool.TabIndex = 0;
            this.lblBool.Text = "VarBool:";
            //
            // lblInt
            //
            this.lblInt.Location = new System.Drawing.Point(40, 80);
            this.lblInt.Name = "lblInt";
            this.lblInt.TabIndex = 1;
            this.lblInt.Text = "VarInt:";
            //
            // lblString
            //
            this.lblString.Location = new System.Drawing.Point(40, 136);
            this.lblString.Name = "lblString";
            this.lblString.TabIndex = 2;
            this.lblString.Text = "VarString:";
            //
            // txtBool
            //
            this.txtBool.Location = new System.Drawing.Point(152, 24);
            this.txtBool.Name = "txtBool";
            this.txtBool.TabIndex = 3;
            this.txtBool.Text = "";
            //
            // txtInt
            //
            this.txtInt.Location = new System.Drawing.Point(152, 80);
            this.txtInt.Name = "txtInt";
            this.txtInt.TabIndex = 4;
            this.txtInt.Text = "";
            //
```

```
            // txtString
            //
            this.txtString.Location = new System.Drawing.Point(152, 136);
            this.txtString.Name = "txtString";
            this.txtString.TabIndex = 5;
            this.txtString.Text = "";
            //
            // btnRead
            //
            this.btnRead.FlatStyle = System.Windows.Forms.FlatStyle.System;
            this.btnRead.Location = new System.Drawing.Point(48, 216);
            this.btnRead.Name = "btnRead";
            this.btnRead.TabIndex = 6;
            this.btnRead.Text = "&Read";
            this.btnRead.Click += new System.EventHandler(this.btnRead_Click);
            //
            // btnWrite
            //
            this.btnWrite.FlatStyle = System.Windows.Forms.FlatStyle.System;
            this.btnWrite.Location = new System.Drawing.Point(168, 216);
            this.btnWrite.Name = "btnWrite";
            this.btnWrite.TabIndex = 7;
            this.btnWrite.Text = "&Write";
            this.btnWrite.Click += new System.EventHandler(this.btnWrite_Click);
            //
            // frmWebServiceTestConsumer
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(292, 266);
            this.Controls.Add(this.lblBool);
            this.Controls.Add(this.lblInt);
            this.Controls.Add(this.lblString);
            this.Controls.Add(this.txtBool);
            this.Controls.Add(this.txtInt);
            this.Controls.Add(this.txtString);
            this.Controls.Add(this.btnRead);
            this.Controls.Add(this.btnWrite);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "frmWebServiceTestConsumer";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "WebServiceTestConsumer";
            this.ResumeLayout(false);

        }
        #endregion

        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.Run(new frmWebServiceTestConsumer());
        }



        #region Event-Methods

        private void btnRead_Click(object sender, System.EventArgs e)
        {
            Read();
        }


        private void btnWrite_Click(object sender, System.EventArgs e)
        {
            Write();
        }


        #endregion


        #region Worker-Methods

        private void Read()
        {
            // DataBuffer for the incoming data
            byte[] abDataBuffer = new byte[84]; //
```

```
1 byte for "PlcVarBool", 2 bytes for "PlcVarInt" and 81 bytes for "PlcVarString" = 84 bytes

            try
            {
                TcWebService.Read(      szAmsNetId, iPort, iIndexGroup, 0, 84, out abDataBuffer);

                // Converts the first byte of the buffer to bool
                bool bVarBool       =       BitConverter.ToBoolean(abDataBuffer, 0);

                // Converts the second and third byte of the buffer to int
                int iVarInt             =       BitConverter.ToInt16(abDataBuffer, 1);

                // Converts the other bytes of the buffer to string
                string szVarString      =       encoder.GetString(abDataBuffer, 3, 81);

                // Writes values into the text-boxes
                txtBool.Text = bVarBool.ToString();
                txtInt.Text = iVarInt.ToString();
                txtString.Text = szVarString;
            }
            catch (SoapException ex)
            {
                MessageBox.Show(ex.Message);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }


        private void Write()
        {
            try
            {
                // byte-arrays for the variables
                byte[]  abDataBool          =       new     byte[1];
                byte[]  abDataInt           =       new     byte[2];
                byte[]  abDataString    =       new     byte[81];


                // Gets values for the text-boxes
                bool    bValueBool          =       Convert.ToBoolean(txtBool.Text);
                int         iValueInt           =       Convert.ToInt16(txtInt.Text);
                string  szValueString   =       txtString.Text;


                // Converting variables to byte-arrays
                abDataBool      =       BitConverter.GetBytes(bValueBool);
                abDataInt       =       BitConverter.GetBytes((Int16)iValueInt);

                encoder.GetBytes(       szValueString,                          // Source
                                    0,                                          // Position of the f
irst character to convert
                                    encoder.GetByteCount(szValueString),    // Gets the length of th
e source
                                    abDataString,                           // Byte-
array to save the conversion to
                                    0);                                         // First byte in the
 array to save the conversion to



                // Writing values to PLC
                TcWebService.Write(szAmsNetId, iPort, iIndexGroup, 0, abDataBool);
                TcWebService.Write(szAmsNetId, iPort, iIndexGroup, 1, abDataInt);
                TcWebService.Write(szAmsNetId, iPort, iIndexGroup, 3, abDataString);
            }
            catch (SoapException ex)
            {
                MessageBox.Show(ex.Message);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }


        #endregion
```

```
    }
}
```

Language / IDE	Extract the sample program
Visual C#	https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488721931/.exe

6.1.2 Consumer in C++ to read and write PLC variables

This sample explains how to create an ADS-WebService-Consumer to read and write PLC-Variables via ADS-HTTP.

Unzip the sample 'https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488723339/.zip'.

1. Creating a WebService-Consumers within the Microsoft Visual C++

Create a new project with the help of the MFC-AppWizard.
Copy th files 'WebServiceConnector.cpp', 'WebServiceConnector.h', 'Base64.cpp' and 'Base64.h' (can be found in the https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488723339/.zip) to your project-folder and add them into the project
Insert the following line to the file in which you want to use the ADS Web Service:

```
#include "WebServiceConnector.h"
```

Before you can read and write you must create an object of the WebServiceConnector and connect to the WebService:

```
CWebServiceConnector webServiceConnector;

webServiceConnector.Connect("http://192.168.0.2.1.1/TcAdsWebService/TcAdsWebService.dll");
```

The address must be the URL of your TwinCAT ADS WebService-DLL-file.

2. Reading PLC -variables

To read a PLC-variable use the "Read"-method of the WebServiceConnector-object:

```
HRESULT Read(char* netId, int port, ULONG indexGroup, ULONG indexOffset, ULONG* pcbData, BYTE* pData
);
```

- **char* netId:** String indicating the AMS-Net-Id of the PLC
- **int port:** Port-number of the Runtime-System
- **ULONG indexGroup:** IndexGroup of the variables to read
- **ULONG indexOffset:** First byte to read
- **ULONG cblen:** Number of bytes to read
- **Byte[] pData:** Byte-array to save the PLC-data into
- **return value:** errorcode [▶ 64]

The 3 methods

```
HRESULT ReadInt (char* netId, int port, ULONG indexGroup, ULONG indexOffset, ULONG cblen, int& ret);
HRESULT ReadBool (char* netId, int port, ULONG indexGroup, ULONG indexOffset, ULONG cblen, bool& ret
);
HRESULT ReadString (char* netId, int port, ULONG indexGroup, ULONG indexOffset, ULONG cblen, CString
& ret);
```

convert the variables to their corresponding type.

Example:

```
bool Boolean;
if (webServiceConnector.ReadBool("192.168.0.2.1.1", 801, 16416, 0, 1, Boolean)!=S_OK)
        MessageBox("Error");
```

3. Writing PLC-variables

```
HRESULT Write(char* netId, int port, ULONG indexGroup, ULONG indexOffset, ULONG cblen, BYTE* pData);
```

- **netId:** String indicating the AMS-Net-Id of the PLC
- **port:** Port-number of the Runtime-System
- **indexGroup:** IndexGroup of the variable
- **indexOffset:** First byte to write to
- **pData:** Byte-array containing the data to write
- **return value:** errorcode [▶ 64]

The 3 methods

```
HRESULT WriteInt(char* netId, int port, ULONG indexGroup, ULONG indexOffset, ULONG cblen, int Value)
;
HRESULT WriteBool(char* netId, int port, ULONG indexGroup, ULONG indexOffset, ULONG cblen, bool Valu
e);
HRESULT WriteString(char* netId, int port, ULONG indexGroup, ULONG indexOffset, ULONG cblen, CString
 Value);
```

convert the variables to a byte-arrays.

Example:

```
if (webServiceConnector.WriteBool("192.168.0.2.1.1", 801, 16416, 0, 1, true)!=S_OK)
        MessageBox("Error");
```

4. Error codes

```
S_OK - No error
E_NO_CON - No connetion to the WebService
E_SEND - Unable to send data to the WebService
E_RESPONSE - Unknown response from the WebService
E_FAILED - Unknown error
```

5. PLC-program, IEC1131 daclaration of the variables:

```
PROGRAM MAIN
VAR
    PlcVarBool    AT %MX0.0   : BOOL := TRUE;
    PlcVarInt     AT %MW1     : INT := 1234;
    PlcVarString  AT %MD3     : STRING := 'Hello Automation';
END_VAR
```

6. C++ Application, the layout:

7. The source-code:

```
void CWebServiceTestConsumerDlg::OnRead()
{
    CWebServiceConnector webServiceConnector;
    webServiceConnector.Connect(WebServiceURL);
    if (webServiceConnector.IsConnected())
```

```
    {
        int Integer;
        bool Boolean;
        CString String;

        int port = 801;
        int group = 16416;
        unsigned long size;
        int offset;


        /* read bool */
        size = 1;
        offset = 0;
        if (webServiceConnector.ReadBool(WebServiceNetId,port,group,offset,size,Boolean) == S_OK)
        {
            m_boolean = Boolean;
        }
        else
        {
            MessageBox("Error");
            return;
        }

        /* read int */
        size = 2;
        offset = 1;
        if (webServiceConnector.ReadInt(WebServiceNetId,port,group,offset,size,Integer) == S_OK)
        {
            m_integer = Integer;
        }
        else
        {
            MessageBox("Error");
            return;
        }

        /* read string */
        size = 81;
        offset = 3;
        if (webServiceConnector.ReadString(WebServiceNetId,port,group,offset,size,String) == S_OK)
        {
            m_string = String;
        }
        else
        {
            MessageBox("Error");
            return;
        }
        UpdateData(false);
    }
    else MessageBox("Error");
}

void CWebServiceTestConsumerDlg::OnWrite()
{
    CWebServiceConnector webServiceConnector;
    webServiceConnector.Connect(WebServiceURL);
    if (webServiceConnector.IsConnected())
    {
        UpdateData(true);

        int port = 801;
        int group = 16416;
        int offset;
        unsigned long size;

        /* write boolean */
        size = 1;
        offset = 0;
        bool boolean = false;
        if (m_boolean==1)
            boolean = true;
        if (webServiceConnector.WriteBool(WebServiceNetId,port,group,offset,size,boolean)!=S_OK)
            MessageBox("Error");

        /* write integer */
        size = 2;
        offset = 1;
```

```
        if (webServiceConnector.WriteInt(WebServiceNetId,port,group,offset,size,m_integer)!=S_OK)
            MessageBox("Error");

        /* write string */
        size = 81;
        offset = 3;

        if (webServiceConnector.WriteString(WebServiceNetId,port,group,offset,size,m_string)!=S_OK)
            MessageBox("Error");
    }
    else MessageBox("Error");
}
```

8. C++ Application, the download:

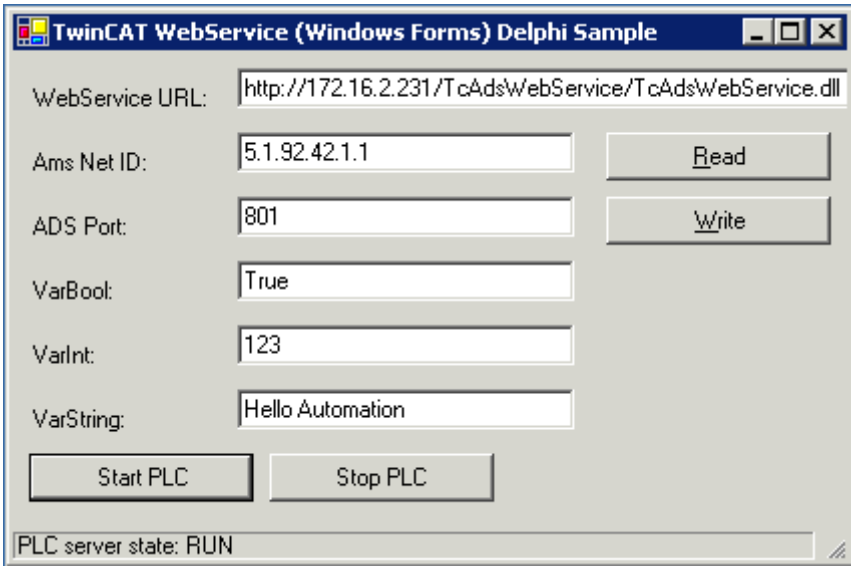
https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488723339/.zip

6.1.3 Consumer in Delphi 8 for .NET to read and write PLC variables

The example explains how to create an ADS-Web Service consumer for reading and writing PLC variables via ADS-HTTP.

Requirements:

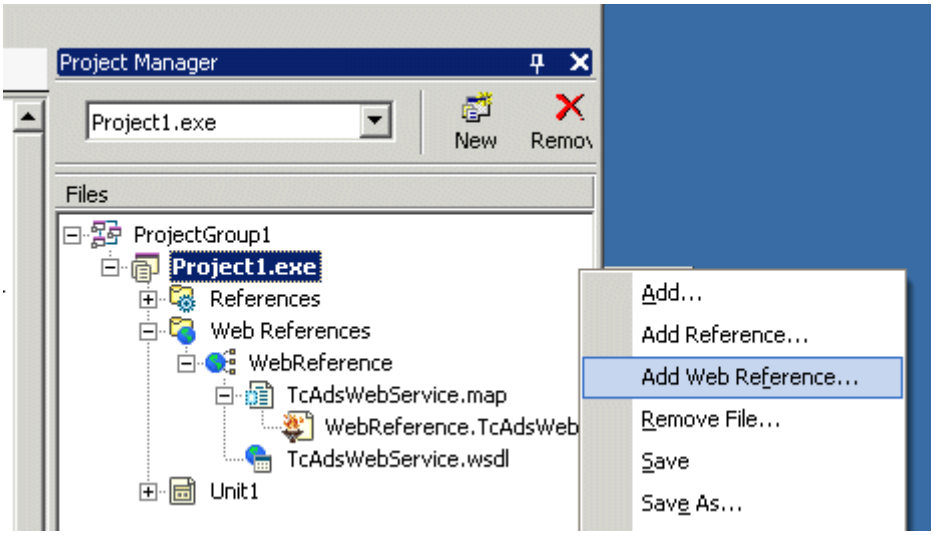
- Delphi 8 for Microsoft .NET framework;
- Web service consumer for Windows XP platform (Windows Forms .NET application);
- Web service server was configured on an Embedded PC: CX9000 configured with Windows CE (ARM).



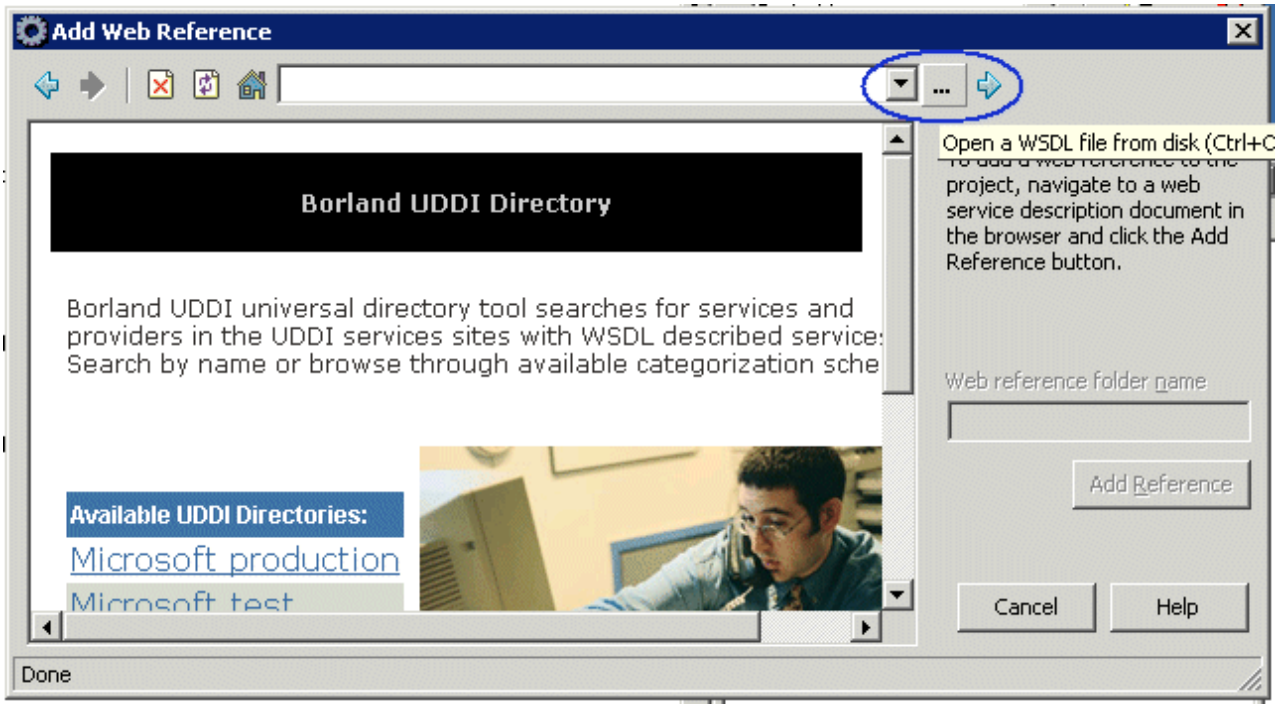
Import into Delphi

The server interface description is published in a WSDL file (Web Service Definition Language). The WSDL description is imported into Delphi and converted by Delphi into Delphi units.

First of all create a new project (Windows Forms application Delphi .NET). Save the project. You can now import the WSDL file. In the Project Manager, right click with the mouse on the project name and select “Add Web Reference”.

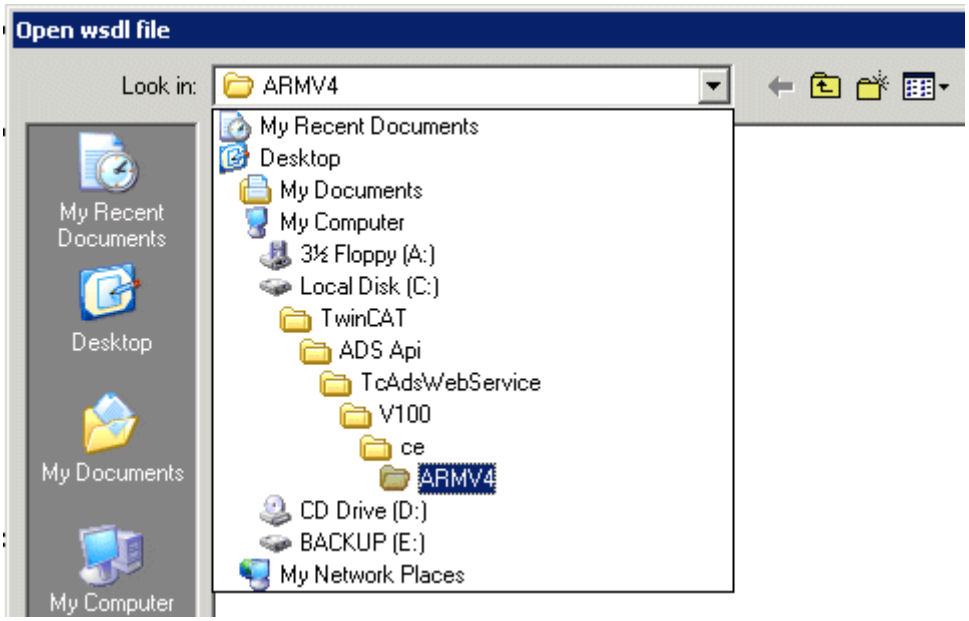


A dialog opens showing Web service directories from IBM, Microsoft etc. However, we want to open the WSDL file from the data storage device.

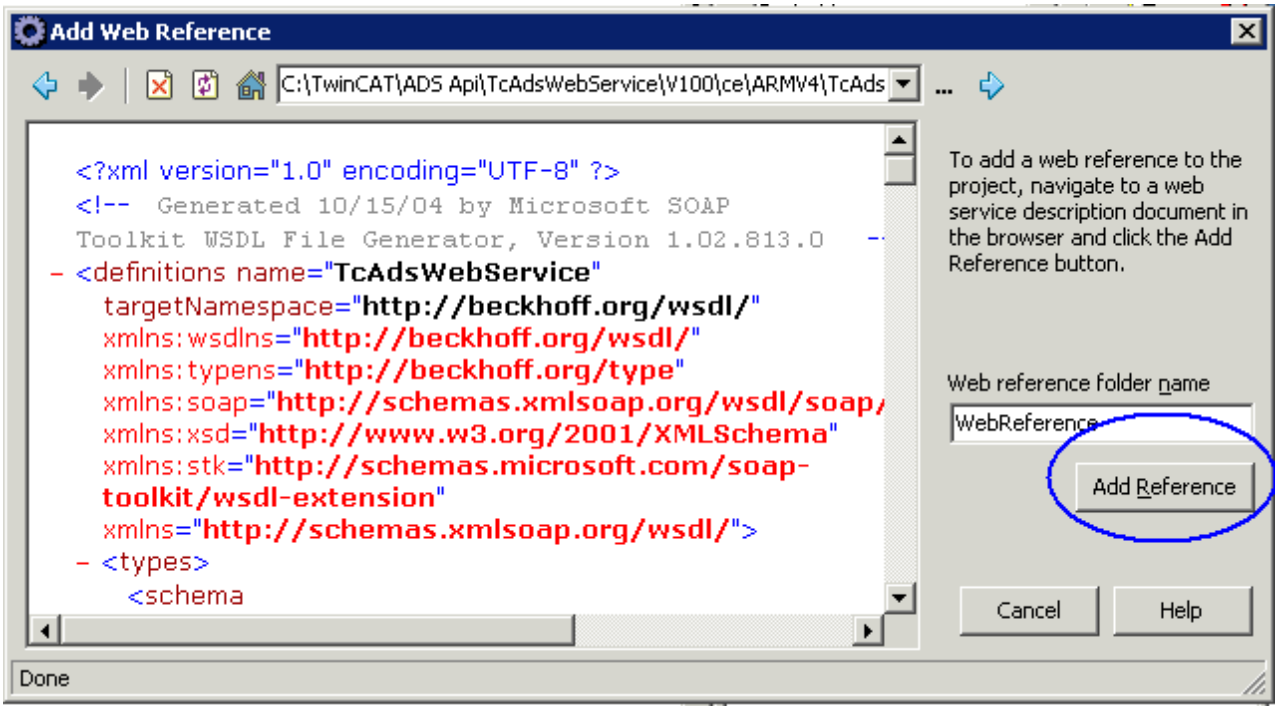


The WSDL files from the TwinCAT ADS Web service can be found in the folder: ..**TwinCAT\Ads API\TcAdsWebService\V100**... Navigate to the appropriate folder and select the WSDL file. In our case it is the WSDL file from the folder for the Windows CE (ARM) target platform.

There is no difference between the WSDL files for the XP/CE (X86 or ARM) platforms. I.e. it is always the same WSDL file. You can integrate the WSDL file from the CE/ARM folder in your consumer project and nonetheless still communicate with an ADS Web Service server on an XP platform.



The contents of the WSDL file are displayed in a dialog box.



Select “Add Web Reference”. The required units are generated automatically and added to the project.

Using the web service

In order to be able to use the web service you must integrate the web service unit in the main window unit. You can take the opportunity at the same time to integrate two further units that you will need in your project later: **System.Text** and **System.Web.Services.Protocols**

```
uses
System.Drawing, System.Collections, System.ComponentModel,
System.Windows.Forms, System.Data,
 WebReference.TcAdsWebService, System.Text,System.Web.Services.Protocols;
```

Now you can create an ADS Web service instance.

```
 private
{ Private Declarations }
TcWebService : WebReference.TcAdsWebService.TcAdsWebService;
```

And in the TwinForm constructor:

```
TcWebService := WebReference.TcAdsWebService.TcAdsWebService.Create();
```

The code completion will help you further. Type in TcWebService with a dot and look at the methods which are available.

We use the following methods in our example:

- Read for reading the PLC variables;
- Write for writing the PLC variables;
- ReadState for reading the PLC server status;
- WriteControl to change the PLC server status;

```
type TArrayOfByte = array of Byte;

procedure Read(netId: string; nPort: Integer; indexGroup: Cardinal; indexOffset: Cardinal; cbL
en: Integer; out ppData: TArrayOfByte);
procedure Write(netId: string; nPort: Integer; indexGroup: Cardinal; indexOffset: Cardinal; pD
ata: TArrayOfByte);
procedure ReadState(netId: string; nPort: Integer; out pAdsState: Integer; out pDeviceState: I
nteger);
procedure WriteControl(netId: string; nPort: Integer; adsState: Integer; deviceState: Integer;
 pData: TArrayOfByte);
```

netId: String with the network address of the TwinCAT target system. In our case the TwinCAT AMS NetID of the PC with the PLC runtime system;

nPort: TwinCAT ADS port number of the PLC runtime system;

indexGroup: Index group of the PLC variables to be read;

indexOffset: Index offset (byte-/bit offset) of the PLC variables to be read;

cbLen: Number of bytes which are to be read;

ppData: Variable in which the read data is saved (dynamic byte array);

pData: Variable with the data to be written (dynamic array);

adsState / pAdsState: PLC server status.

deviceState / pDeviceState: PLC device status;

The PLC application

```
PROGRAM MAIN
VAR
    VarBool    AT %MX0.0:  BOOL := TRUE;
    VarInt     AT %MB1:    INT := 123;
    VarString  AT %MB3:    STRING := 'Hello Automation';
END_VAR
```

Read PLC variables

```
procedure TWinForm.ReadData();
var cbLen, nIG, nIO   : Integer;
    data    : TArrayOfByte;
begin
  try
    // read BOOL at adress AT%MX0.0
    nIG   := $4021;
    nIO   := 0;
    cbLen := SizeOf(varBool);
    TcWebService.Read( sNetID, iPort, nIG, nIO, cbLen, data );
    varBool     := BitConverter.ToBoolean(data, 0);
    // read INT at adress AT%MB1
    nIG   := $4020;
    nIO   := 1;
    cbLen := SizeOf(varInt);
    TcWebService.Read( sNetID, iPort, nIG, nIO, cbLen, data );
    varInt      := BitConverter.ToInt16(data, 0);
    // read STRING at adress AT%MB3
    nIG   := $4020;
    nIO   := 3;
```

```
    cbLen := 81;
    TcWebService.Read( sNetID, iPort, nIG, nIO, cbLen, data );
    varString := System.Text.Encoding.Default.GetString(data,0, Length(data));
  except
    On E:SoapException do
      HandleSoapException(E.Message, E.HelpLink, E.Detail.InnerText);
  end;
end;
```

Write PLC variables

```
procedure TWinForm.WriteData();
var data    : TArrayOfByte;
    cbLen, nIG, nIO : Integer;
begin
  try
    // write BOOL at adress AT%MX0.0
    nIG   := $4021;
    nIO   := 0;
    data  := BitConverter.GetBytes(varBool);
    TcWebService.Write( sNetID, iPort, nIG, nIO, data );
    // write INT at adress AT%MB1
    nIG   := $4020;
    nIO   := 1;
    data  := BitConverter.GetBytes(varInt);
    TcWebService.Write( sNetID, iPort, nIG, nIO, data );
    // write STRING at adress AT%MB3
    nIG   := $4020;
    nIO   := 3;
    cbLen := System.Text.Encoding.Default.GetByteCount(varString);
    // Allocate one byte more! We need it to write the string end delimiter!.
    SetLength(data, cbLen + 1);
    System.Text.Encoding.Default.GetBytes( varString, 0, cbLen, data, 0);
    data[cbLen] := 0;
    TcWebService.Write( sNetID, iPort, nIG, nIO, data );
  except
    On E:SoapException do
      HandleSoapException(E.Message, E.HelpLink, E.Detail.InnerText);
  end;
end;
```

Read PLC status

```
procedure TWinForm.ReadState();
var adsState, deviceState : Integer;
sState : String;
begin
  try
    TcWebService.ReadState( sNetID, iPort, adsState, deviceState );
    case adsState of
      PLC_ADSSTATE_RUN:
    sState := 'RUN';
      PLC_ADSSTATE_STOP:
    sState := 'STOP';
    else
      sState := 'OTHER';
    end;
    StatusBar1.Panels.Item[0].Text := 'PLC server state: ' + sState;
  except
    On E:SoapException do
      HandleSoapException(E.Message, E.HelpLink, E.Detail.InnerText);
  end;
end;
```

Start/stop PLC

```
procedure TWinForm.WriteControl(adsState : Integer);
begin
  try
    TcWebService.WriteControl( sNetID, iPort, adsState, 0, nil );
  except
    On E:SoapException do
      HandleSoapException(E.Message, E.HelpLink, E.Detail.InnerText);
  end;
end;
```

Language / IDE	Extract the sample program
Delphi 8 for .NET	https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488730379/.exe

6.1.4 Consumer in Java to read and write PLC variables

This sample explains how to create an ADS-WebService-Consumer to read and write PLC-Variables via ADS-HTTP.

Unzip the sample 'https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488724747/.zip'.

Copy the files "TcAdsSOAP.java" and "Base64.java" to the folder of your Java-application and compile them. Add the following line to the Java-file in which you want to use the ADS WebService:

```
TcAdsSOAP tcSoap = new TcAdsSOAP("http://192.168.0.2/TcAdsWebService/TcAdsWebService.dll");
```

The address must be the URL of your TwinCAT ADS WebService-DLL-file.

reading PLC-variables

To read PLC-variables use the following method of the TcAdsSoap-object:

```
DataInputStream SOAPCall(String netId, int port, long group, long offset, int cblen)
```

- **String netId:** String indicating the AMS-Net-Id of the PLC
- **int port:** Port-number of the Runtime-System
- **long indexGroup:** IndexGroup of the variables to read
- **long indexOffset:** First byte to read
- **int cblen:** Number of bytes to read
- **return value:** DataInputStream with the read data, throws exception on failure

The 3 methods

```
short ReadInt(String netId, int port, long group, long offset)
boolean ReadBool(String netId, int port, long group, long offset)
String ReadString(String netId, int port, long group, long offset, int cblen)
```

convert the variables to their corresponding type.

Example:

```
boolean Bool1;
try
{
    Bool1 = tcSoap.ReadBool("192.168.0.2.1.1", 801, 16416, 0)
}
catch (Exception e)
{
    System.out.println("Error");
}
```

Write PLC-variables

```
boolean SOAPWrite(String netId, int port, long group, long offset, byte[] data)
```

- **String netId:** String indicating the AMS-Net-Id of the PLC
- **int port:** Port-number of the Runtime-System
- **long indexGroup:** IndexGroup of the variable
- **long indexOffset:** First byte to write to
- **byte[] pData:** Byte-array containing the data to write
- **return value:** "true" if successful, throws exception on failure

The 3 methods

```
boolean WriteBool(String netId, int port, long group, long offset, boolean data)
boolean WriteInt(String netId, int port, long group, long offset, int data)
boolean WriteString(String netId, int port, long group, long offset, String data)
```

convert the variables to a byte-arrays.

Example:

```
try
{
    tcSoap.WriteBool("192.168.0.2.1.1", 801, 16416, true)
}
catch (Exception e)
{
    System.out.println("Error");
}
```

PLC-program, IEC1131 daclaration of the variables:

```
PROGRAM MAIN
VAR
    PlcVarBool    AT %MX0.0  : BOOL := TRUE;
    PlcVarInt     AT %MW1    : INT := 1234;
    PlcVarString  AT %MD3    : STRING := 'Hello Automation';
END_VAR
```

The source-code:

```
public class AdSWebServiceSample
{
    public static void main(String args[])
    {
        TcAdsSOAP tcSoap = new TcAdsSOAP("http://192.168.0.2/TcAdsWebService/TcAdsWebService.dll");
        try
        {
            String netId = "192.168.0.2.1.1";
            int port = 801;
            int offset = 16416;
            /* write bool */
            boolean testBool = true;
            if (tcSoap.WriteBool(netId,port,offset,0,testBool))
                System.out.println("Boolean "+testBool+" Written");

            /* write int */
            int testInt = 1234;
            if (tcSoap.WriteInt(netId,port,offset,1,testInt))
                System.out.println("Integer "+testInt+" Written");

            /* write string */
            String testString = "Hello Automation";
            if (tcSoap.WriteString(netId,port,offset,3,testString))
                System.out.println("String "+testString+" Written");



            /* read bool */
            System.out.println(String.valueOf(tcSoap.ReadBool(netId,port,offset,0) ));

            /* read int */
            System.out.println(String.valueOf(tcSoap.ReadInt(netId,port,offset,1) ));

            /* read string */
            System.out.println(tcSoap.ReadString(netId,port,offset,3,81));


        }
        catch(Exception ex)
        {
            System.out.println(String.valueOf(ex));
        }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488724747/.zip

6.1.5 ASP.NET 2.0 AJAX

Microsoft ASP.NET 2.0 AJAX Extensions is the new name for the ASP.NET technology code-named "Atlas".

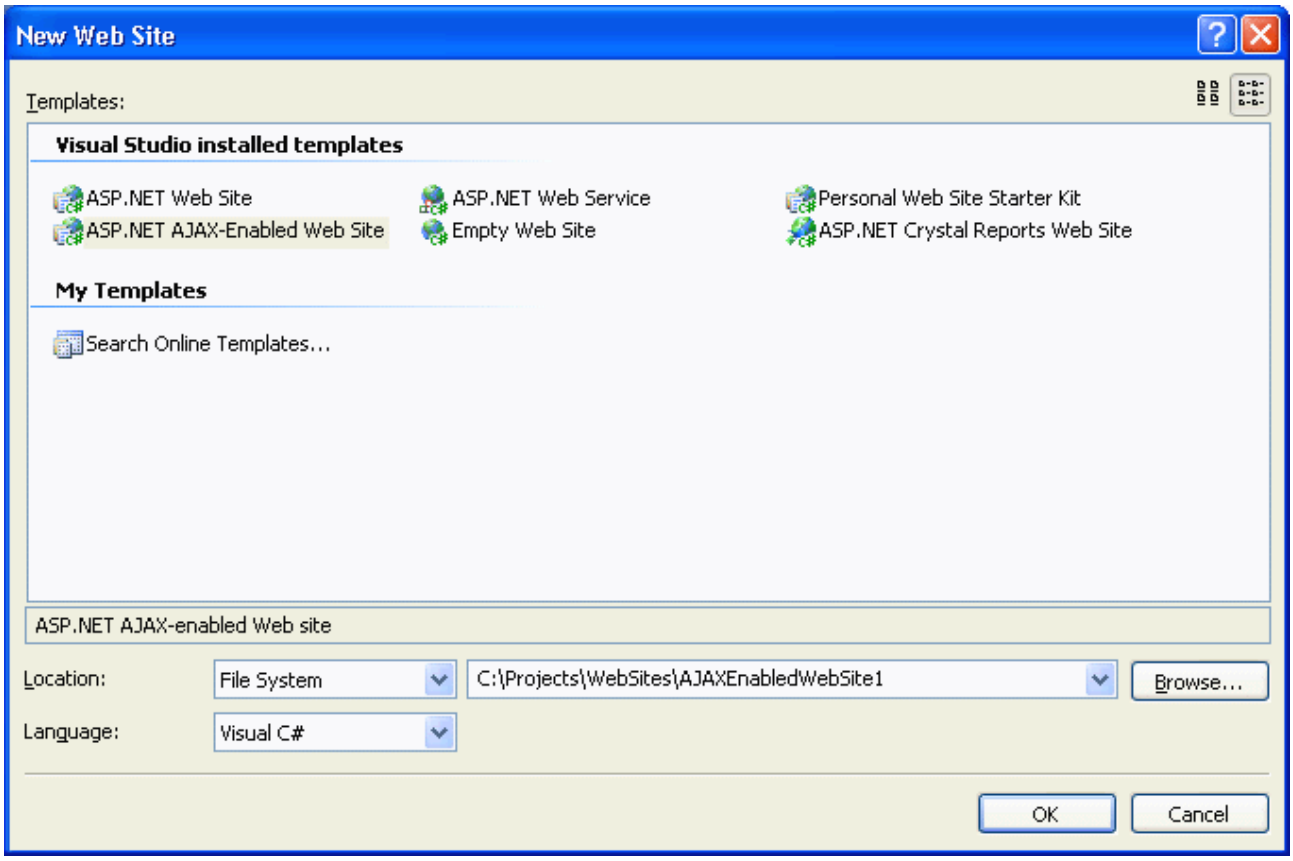
System Requirements

- **Supported Operating Systems**: Windows Server 2003; Windows Vista; Windows XP
- **Required Software:**
  - Microsoft .NET Framework Version 2.0
    Link: http://msdn.microsoft.com/netframework/downloads/updates/default.aspx
  - Microsoft Internet Explorer 5.01 (or later)
  - Visual Studio 2005
  - Framework "ASP.NET 2.0 AJAX Extensions" (free)
    Link : http://ajax.asp.net/
  - Microsoft ASP.NET AJAX

Get Started with ASP.NET 2.0 AJAX in VS2005?

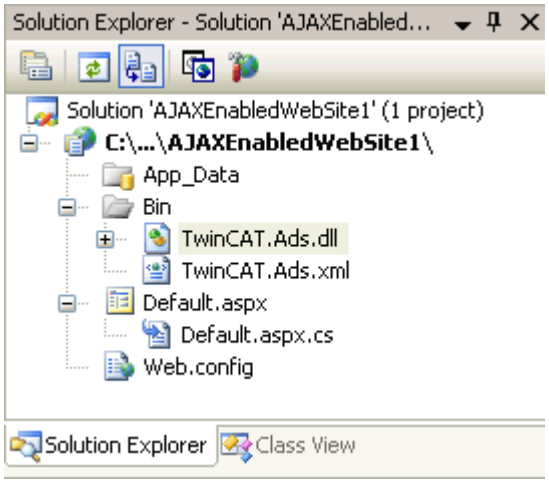
Step 1. Create a new project

Open Visual Studio 2005, choose under *File* , *New* the pption *Website...* Now you create a new website.
Choose under the "Templates" the "ASP.NET AJAX-Enabled Web Site" template.
Choose the source code language: Visual C#



Step 2. Add the reference:

Add the reference of TwinCAT.Ads.dll to VS2005 -> *Menu -> Website -> Add Reference....*
The default installation path for .NET Framework 2.0 of TwinCAT.Ads.dll is C:\TwinCAT\ADS Api\.NET\v2.0.50727\TwinCAT.ADS.dll
Please work with the latest greatest Version.



Step 3. Add AJAX component:

Add the *Timer* and the *UpdatePanel* of AJAX Extensions to the web file "Default.aspx" from Toolbox of Visual Studio 2005.

4. Edit properties:

Change *Interval* of the *Timer1*:

```
Interval = "3000"
Change the option UpdateMode of the UpdatePanel1:
```

```
UpdateMode = "Conditional"
```

5. Read PLC variables:

Use the TcAdsClient.Read() to read the variables from ADS device. Reads data synchronously from an ADS device and writes it to the given stream.

```
TcAdsClient.Read(int indexGroup, int indexOffset, AdsStream dataStream, int offset, int length);
```

Parameters:

- **indexGroup:** - Contains the index group number of the requested ADS service.
- **indexOffset:** - Contains the index offset number of the requested ADS service.

- **dataStream:** - Stream that receives the data.
- **offset:** - Offset of the data in dataStream.
- **length:** - Length of the data in dataStream.

Step 6. ASP.NET 2.0 AJAX Sample 01

Default.aspx

```
<%@ Page Language="C#"AutoEventWireup="true"CodeFile="Default.aspx.cs"Inherits="_Default"%>

<!DOCTYPE html PUBLIC"-//W3C//DTD XHTML 1.1//EN""http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>ASP.NET 2.0 AJAX Sample 01</title>
</head>
<body>
    <form id="form1"method="post"runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1"runat="server" />
        <asp:Timer ID="Timer1"runat="server"Interval="3000"OnTick="Timer1_Tick">
        </asp:Timer>

        <!-- ASP.NET AJAX UpdatePanel1 - Only the panel contet is updated. -->
        <asp:UpdatePanel ID="UpdatePanel1"runat="server"UpdateMode="Conditional">
        <ContentTemplate>

            UpdatePanel content refreshed at <%=DateTime.Now.ToString() %>

            <asp:Label ID="lblPlcVarBool01"runat="server"Text="PlcVarBool:"Style="z-
index: 101; left: 10px; position: absolute; top: 50px"></asp:Label>
            <asp:Label ID="lblPlcVarBool02"runat="server"Text="-"BackColor="Black"ForeColor="White"S
tyle="z-index: 101; left: 100px; position: absolute; top: 50px"></asp:Label>

            <asp:Label ID="lblPlcVarInt01"runat="server"Text="PlcVarInt:"Style="z-
index: 101; left: 10px; position: absolute; top: 100px"></asp:Label>
            <asp:Label ID="lblPlcVarInt02"runat="server"Text="-"BackColor="Black"ForeColor="White"St
yle="z-index: 101; left: 100px; position: absolute; top: 100px"></asp:Label>

            <asp:Label ID="lblMessage01"runat="server"Text="Message:"Style="z-
index: 101; left: 10px; position: absolute; top: 200px"></asp:Label>
            <asp:Label ID="lblMessage02"runat="server"Text="-"Style="z-
index: 101; left: 100px; position: absolute; top: 200px"></asp:Label>

            <asp:Button ID="btnRefreshAll"runat="server"Text="Refresh All"Width="160px"OnClick="btnR
efreshAll_Click"Style="z-index: 201; left: 207px; position: absolute; top: 250px"/>

            <asp:Label ID="lblPlcVarString01"runat="server"Text="PlcVarString:"Style="z-
index: 101; left: 10px; position: absolute; top: 150px"></asp:Label>
            <asp:Label ID="lblPlcVarString02"runat="server"Text="-"BackColor="Black"ForeColor="White
"Style="z-index: 101; left: 100px; position: absolute; top: 150px"></asp:Label>

        </ContentTemplate>
        <Triggers>
            <asp:AsyncPostBackTrigger ControlID="Timer1" />
        </Triggers>
        </asp:UpdatePanel>
    </div>
    </form>
</body>
</html>
```

Default.aspx.cs

Change the AMS NET ID and AdsServerPort settings

You must change the AMS NET ID and AdsServerPort to your local project settings.

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
```

```
using System.Web.UI.HtmlControls;
using System.Drawing;
using TwinCAT.Ads;
using System.IO;


public partial class_Default : System.Web.UI.Page
{
    // ADS Kommunication with ADS deviceTcAdsClient tcAds = newTcAdsClient();

    // AMS NET ID <-
 Change this AMS NET ID for your TwinCAT Configurationstring strAmsNetId = "192.168.0.2.1.1";

    // AMS Port (801, 811, 821, 831) <-
 Change this AMS Port for your TwinCAT Configurationint iAdsServerPort = 801;

    protected void Page_Load(object sender, EventArgs e)
    {

    }

    public void RemoteStatusRead()
    {
    try
    {
        tcAds.Connect(strAmsNetId, iAdsServerPort);
        tcAds.Timeout = 200;

        AdsStream dataStream = newAdsStream(171);
        BinaryReader binRead = newBinaryReader(dataStream);

        // read bool variable with IndexGroup and IndexOffset
        tcAds.Read(16417, 0, dataStream, 0, 1);
        // read int variable with IndexGroup and IndexOffset
        tcAds.Read(16416, 1, dataStream, 1, 2);
        // read string variable with IndexGroup and IndexOffset
        tcAds.Read(16416, 5, dataStream, 3, 168);

        bool bVarBool = false;
        byte b = binRead.ReadByte();
        if (b == 0) bVarBool = false;
        else bVarBool = true;

        if (bVarBool == true)
        {
        lblPlcVarBool02.Text = "TRUE";
        lblPlcVarBool02.BackColor = Color.Blue;
        lblPlcVarBool02.ForeColor = Color.White;
        }
        else
        {
        lblPlcVarBool02.Text = "FALSE";
        lblPlcVarBool02.BackColor = Color.Black;
        lblPlcVarBool02.ForeColor = Color.White;
        }

        int iData = 0;
        iData = binRead.ReadByte();

        if (iData != 0)
        {
        lblPlcVarInt02.Text = iData.ToString();
        }
        else
        {
        lblPlcVarInt02.Text = "-";
        }

        char [] strData = new char[160];
        strData = binRead.ReadChars(160);

        if (strData != null)
        {
        lblPlcVarString02.Text = "";
        for (int i = 1; i < 160; i++)
        {
            lblPlcVarString02.Text += strData[i];
        }
        }
        else
```

```
        {
        lblPlcVarString02.Text = "-";
        }
        lblMessage02.Text = "";
    }
    catch(Exception ex)
    {
        lblMessage02.Text = ex.Message;
    }
    }

   protected void btnRefreshAll_Click(object sender, EventArgs e)
    {
    RemoteStatusRead();

    // update content of the UpdatePanel1
    UpdatePanel1.Update();
    }

    protected void Timer1_Tick(object sender, EventArgs e)
    {
    RemoteStatusRead();

    // update content of the UpdatePanel1
    UpdatePanel1.Update();
    }
}
```

Step 7. PLC Sample

```
PROGRAM MAIN
VAR
PlcVarBool    AT%MX0.0  : BOOL := TRUE;
PlcVarInt     AT%MW1    : INT := 0;
PlcVarIntMax  AT%MW3    : INT := 1000;
PlcVarString  AT%MD5    : STRING(20);
TON1: TON;
bool1: BOOL;
time1: TIME;
END_VAR
```

```
PlcVarInt := PlcVarInt + 1;
IF PlcVarInt > PlcVarIntMax THEN
PlcVarInt := 0;
PlcVarBool := NOT PlcVarBool;
END_IF

TON1(IN:=TRUE , PT:=t#5m , Q=>bool1 , ET=>time1 );
IF bool1 THEN
TON1(IN:=FALSE , PT:= , Q=> , ET=> );
END_IF

PlcVarString := TIME_TO_STRING(time1);
```

Step 8. Download samples

https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488726155/.zip
https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488727563/.zip
https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488728971/.zip

Step 9. Configuration of the Microsoft Internet Information Services (IIS)

Open the IIS:

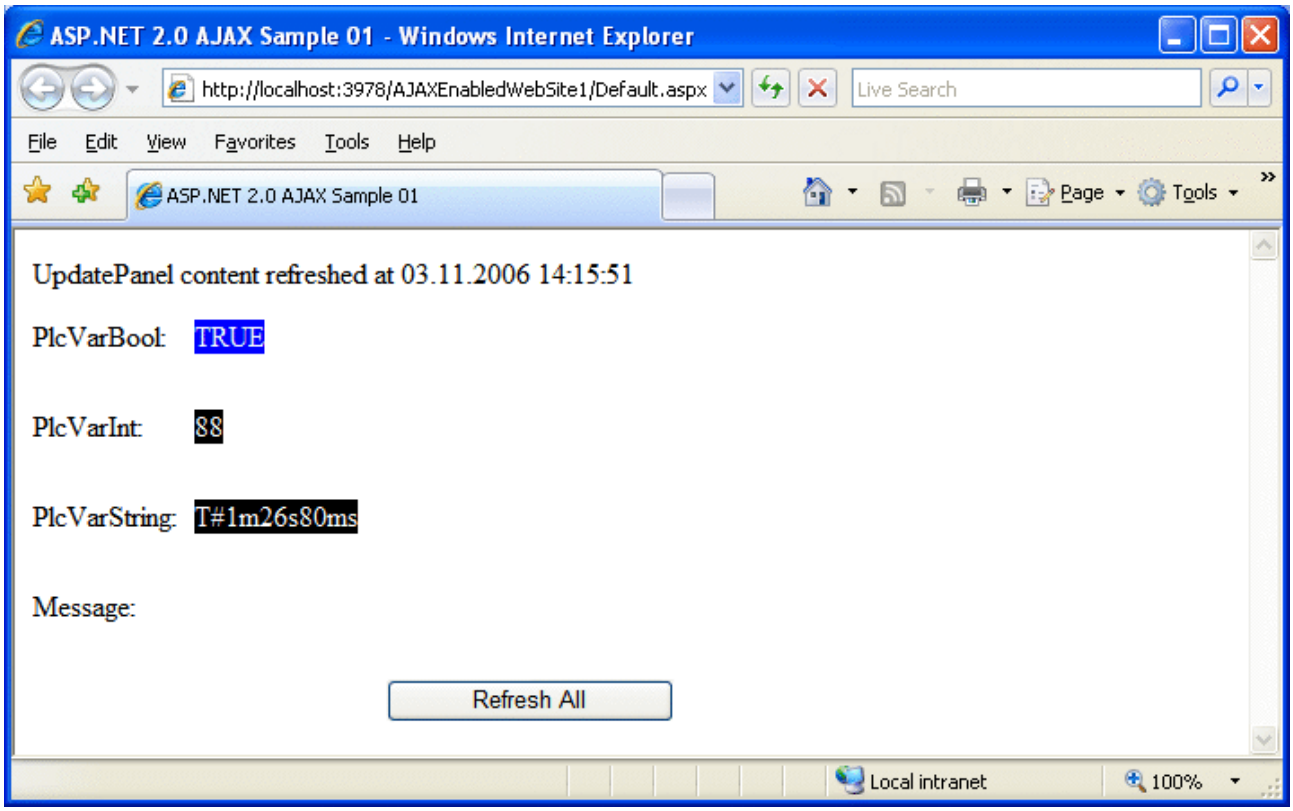
- Start -> Control Panel -> Administrative Tools -> Internet Information Services
- Add a new virtual Directory for a new website: IIS -> (local computer) -> Web Sites -> Default Web Sites -> New -> Virtual Directory...

Settings for IIS:

- Alias: AjaxSample01
- Directory: C:\Projects\WebSites\AJAXEnabledWebSite1
- Access Permisions: Choose "Read", "Run scripts" and "Execute"

- Important: Click to Node "AjaxSample01" and change the IIS Properties "ASP.NET" of the Web site to Version 2.0

Step 10. Demo of the AJAX Website Sample Microsoft IE

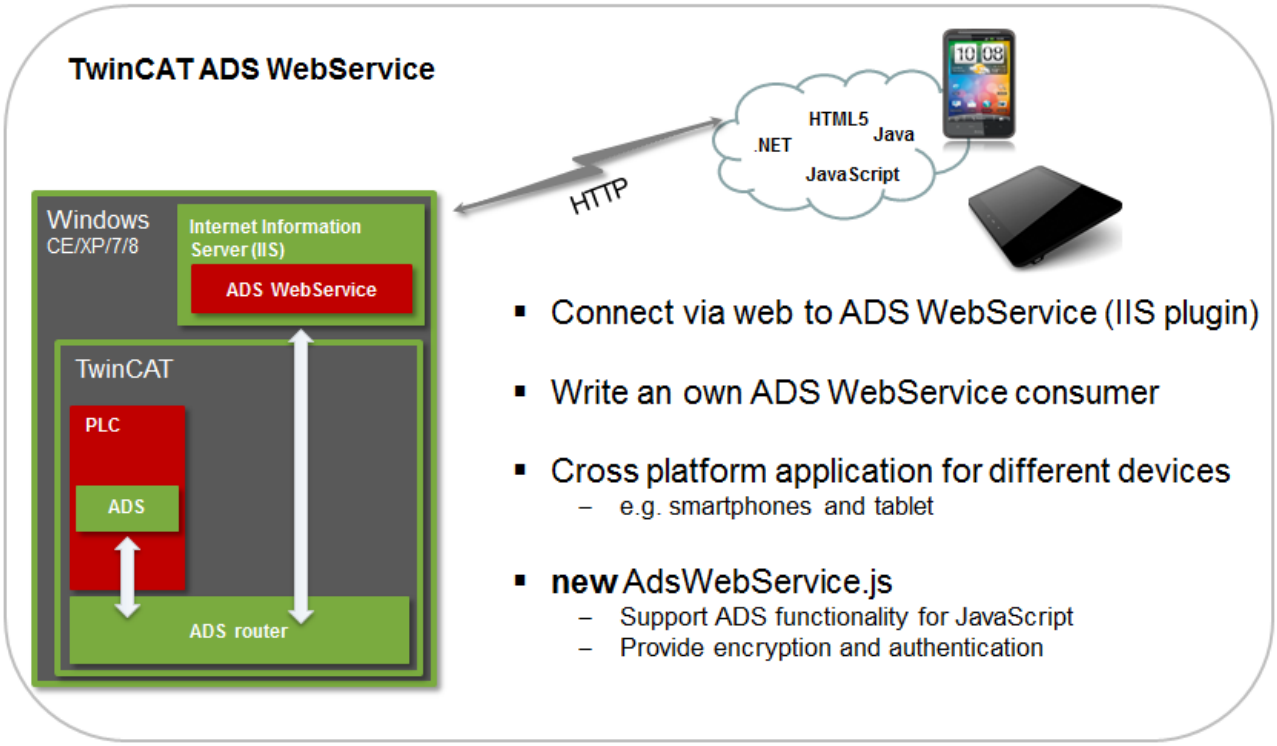


Browse to the website with the "Default.aspx" in IIS
or open the link http://localhost/AjaxSample01/Default.aspx with Internet Explorer
(p.E. http://192.168.0.2/AjaxSample01/Default.aspx or http://<ip-of-PC>/AjaxSample01/Default.aspx).

This example can be used only with access by a user with an instance.

6.2 Samples: TcAdsWebService.js

The TcAdsWebService.js JavaScript Library provides objects for an easier usage of the TcAdsWebService to cross platform applications.



Samples
Cyclic reading of multiple variables with sumcommando [▶ 79]
Writing multiple variables with sumcommando [▶ 84]
Description Accessing an array in the PLC Event driven reading Reading and writing string variables [▶ 88]

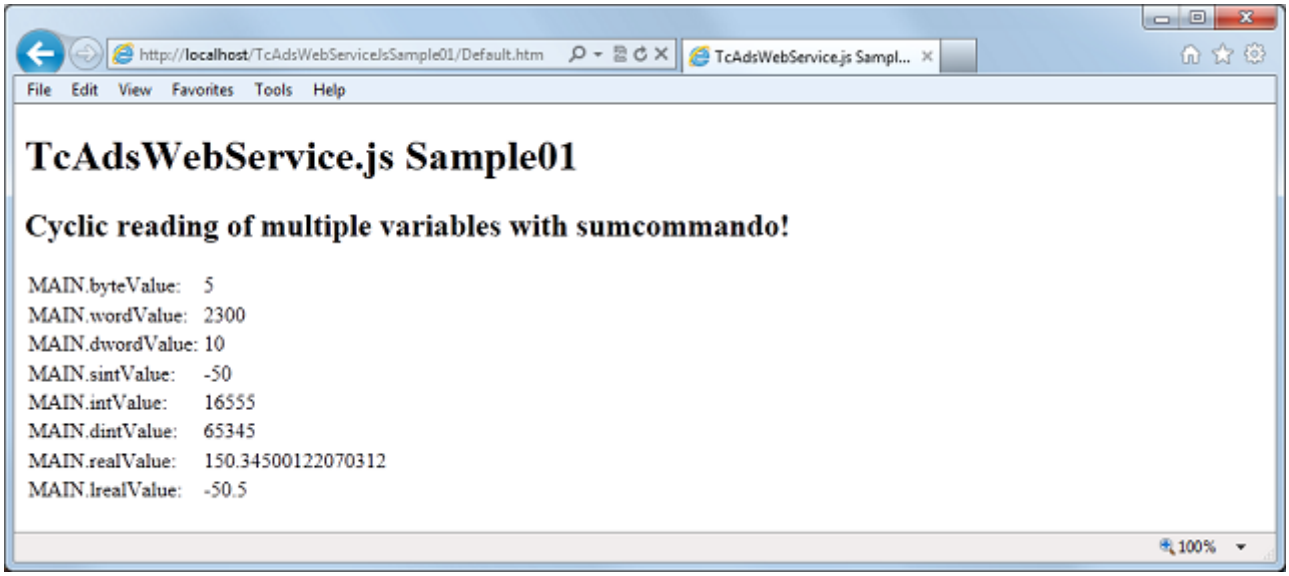
6.2.1 Cyclic reading of multiple variables with sumcommando

Description:

This sample shows how to read multiple variables cyclic by use of a read sumcommando.

Transporting bigger data packets

Generally, HTTP-traffic is more efficient by transporting bigger data packets instead of many small packets. Therefore please use ADS sum commandos to collect your data.



ToDo:

- Download the https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488970251/.zip
- Start the PLC project
- Copy the default.htm and TcAdsWebService.js to your configured virtual directory
- Adjust NETID, PORT and SERVICE_URL if you use the TcAdsWebservice of a remote pc

```
<!DOCTYPE html>
<html>
<head>
    <title>TcAdsWebService.js Sample01</title>
    <script type="text/javascript" src="TcAdsWebService.js"></script>
    <script type="text/javascript">
    (function () {

        var NETID = ""; // Empty string for local machine;
        var PORT = "801"; // TC2 PLC Runtime = 801, TC3 PLC Runtime = 851
        var SERVICE_URL = "http://localhost/TcAdsWebService/
TcAdsWebService.dll"; // HTTP path to the TcAdsWebService;

        var client = new TcAdsWebService.Client(SERVICE_URL, null, null);

        var general_timeout = 500;

        var readLoopID = null;
        var readLoopDelay = 500;

        var readSymbolValuesData = null;

        // Array of symbol names to read;
        var handlesVarNames = [
            "MAIN.byteValue",
            "MAIN.wordValue",
            "MAIN.dwordValue",
            "MAIN.sintValue",
            "MAIN.intValue",
            "MAIN.dintValue",
            "MAIN.realValue",
            "MAIN.lrealValue"
        ];

        // Occurs if the window has loaded;
        window.onload = (function () {

        // Create sumcommando for reading twincat symbol handles by symbol name;
        var handleswriter = new TcAdsWebService.DataWriter();

        // Write general information for each symbol handle to the TcAdsWebService.DataWriter
object;
        for (var i = 0; i < handlesVarNames.length; i++) {
            handleswriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolHandleByNam
e);
            handleswriter.writeDINT(0);
            handleswriter.writeDINT(4); // Expected size; A handle has a size of 4 byte;
            handleswriter.writeDINT(handlesVarNames[i].length); // The length of the symbol na
me string;
        }

        // Write symbol names after the general information to the TcAdsWebService.DataWriter
object;
        for (var i = 0; i < handlesVarNames.length; i++) {
            handleswriter.writeString(handlesVarNames[i]);
        }

        // Send the list-read-
write command to the TcAdsWebService by use of the readwrite function of the TcAdsWebService.C
lient object;
        client.readwrite(
            NETID,
            PORT,
            0xF082,     // IndexGroup = ADS list-read-
write command; Used to request handles for twincat symbols;
            handlesVarNames.length, // IndexOffset = Count of requested symbol handles;
            (handlesVarNames.length * 4) + (handlesVarNames.length * 8), // Length of requeste
```

```
d data + 4 byte errorcode and 4 byte length per twincat symbol;
            handleswriter.getBase64EncodedData(),
            RequestHandlesCallback,
            null,
            general_timeout,
            RequestHandlesTimeoutCallback,
            true);

        });

        // Occurs if the readwrite for the sumcommando has finished;
        var RequestHandlesCallback = (function (e, s) {

        if (e && e.isBusy) {
            // HANDLE PROGRESS TASKS HERE;
            var message = "Requesting handles...";
            td_byteValue.innerHTML = message;
            td_wordValue.innerHTML = message;
            td_dwordValue.innerHTML = message;
            td_sintValue.innerHTML = message;
            td_intValue.innerHTML = message;
            td_dintValue.innerHTML = message;
            td_realValue.innerHTML = message;
            td_lrealValue.innerHTML = message;
            // Exit callback function because request is still busy;
            return;
        }

        if (e && !e.hasError) {

            // Get TcAdsWebService.DataReader object from TcAdsWebService.Response object;
            var reader = e.reader;

            // Read error code and length for each handle;
            for (var i = 0; i < handlesVarNames.length; i++) {

            var err = reader.readDWORD();
            var len = reader.readDWORD();

            if (err != 0) {
                div_log.innerHTML = "Handle error!";
                return;
            }

            }

            // Read handles from TcAdsWebService.DataReader object;
            var hByteValue = reader.readDWORD();
            var hWordValue = reader.readDWORD();
            var hDwordValue = reader.readDWORD();
            var hSintValue = reader.readDWORD();
            var hIntValue = reader.readDWORD();
            var hDintValue = reader.readDWORD();
            var hRealValue = reader.readDWORD();
            var hLrealValue = reader.readDWORD();

            // Create sum commando to read symbol values based on the handle
            var readSymbolValuesWriter = new TcAdsWebService.DataWriter();

            //  "MAIN.byteValue" // BYTE
            readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolVa
lueByHandle); // IndexGroup
            readSymbolValuesWriter.writeDINT(hByteValue); // IndexOffset = The target handle
            readSymbolValuesWriter.writeDINT(1); // size to read

            //  "MAIN.wordValue" // WORD
            readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolVa
lueByHandle); // IndexGroup
            readSymbolValuesWriter.writeDINT(hWordValue); // IndexOffset = The target handle
            readSymbolValuesWriter.writeDINT(2); // size to read

            //  "MAIN.dwordValue" // DWORD
            readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolVa
lueByHandle); // IndexGroup
            readSymbolValuesWriter.writeDINT(hDwordValue); // IndexOffset = The target handle
            readSymbolValuesWriter.writeDINT(4); // size to read

            //  "MAIN.sintValue" // SINT
            readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolVa
lueByHandle); // IndexGroup
```

```
            readSymbolValuesWriter.writeDINT(hSintValue); // IndexOffset = The target handle
            readSymbolValuesWriter.writeDINT(1); // size to read

            //  "MAIN.intValue" // INT
            readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolVa
lueByHandle); // IndexGroup
            readSymbolValuesWriter.writeDINT(hIntValue); // IndexOffset = The target handle
            readSymbolValuesWriter.writeDINT(2); // size to read

            //  "MAIN.dintValue" // DINT
            readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolVa
lueByHandle); // IndexGroup
            readSymbolValuesWriter.writeDINT(hDintValue); // IndexOffset = The target handle
            readSymbolValuesWriter.writeDINT(4); // size to read

            //  "MAIN.realValue" // REAL
            readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolVa
lueByHandle); // IndexGroup
            readSymbolValuesWriter.writeDINT(hRealValue); // IndexOffset = The target handle
            readSymbolValuesWriter.writeDINT(4); // size to read

            //  "MAIN.lrealValue" // LREAL
            readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolVa
lueByHandle); // IndexGroup
            readSymbolValuesWriter.writeDINT(hLrealValue); // IndexOffset = The target handle
            readSymbolValuesWriter.writeDINT(8); // size to read

            // Get Base64 encoded data from TcAdsWebService.DataWriter;
            readSymbolValuesData = readSymbolValuesWriter.getBase64EncodedData();

            // Start cyclic reading of symbol values;
            readLoopID = window.setInterval(ReadLoop, readLoopDelay);

        } else {

            if (e.error.getTypeString() == "TcAdsWebService.ResquestError") {
            // HANDLE TcAdsWebService.ResquestError HERE;
            div_log.innerHTML = "Error: StatusText = " + e.error.statusText + " Status: " + e.
error.status;
            }
            else if (e.error.getTypeString() == "TcAdsWebService.Error") {
            // HANDLE TcAdsWebService.Error HERE;
            div_log.innerHTML = "Error: ErrorMessage = " + e.error.errorMessage + " ErrorCode:
 " + e.error.errorCode;
            }

        }

        });

        // Occurs if the readwrite for the sumcommando to request symbol handles runs into tim
eout;
        var RequestHandlesTimeoutCallback = (function () {
        // HANDLE TIMEOUT HERE;
        div_log.innerHTML = "Requuest handles timeout!";
        });

        // Interval callback for cyclic reading;
        var ReadLoop = (function () {

        // Send the read-read-
write command to the TcAdsWebService by use of the readwrite function of the TcAdsWebService.C
lient object;
        client.readwrite(
            NETID,
            PORT,
            0xF080, // 0xF080 = Read command;
            8, // IndexOffset = Variables count;
            26 + (8 * 4), // Length of requested data + 4 byte errorcode per variable;
            readSymbolValuesData,
            ReadCallback,
            null,
            general_timeout,
            ReadTimeoutCallback,
            true);

        });

        // Occurs if the read-read-write command has finished;
        var ReadCallback = (function (e, s) {
```

```
        if (e && e.isBusy) {
            // HANDLE PROGRESS TASKS HERE;
            // Exit callback function because request is still busy;
            return;
        }

        if (e && !e.hasError) {

            var reader = e.reader;

            // Read error codes from begin of TcAdsWebService.DataReader object;
            for (var i = 0; i < handlesVarNames.length; i++) {
            var err = reader.readDWORD();
            if (err != 0) {
                div_log.innerHTML = "Symbol error!";
                return;
            }
            }

            //  "MAIN.byteValue" // BYTE
            var byteValue = reader.readBYTE();

            //  "MAIN.wordValue" // WORD
            var wordValue = reader.readWORD();

            //  "MAIN.dwordValue" // DWORD
            var dwordValue = reader.readDWORD();

            //  "MAIN.sintValue" // SINT
            var sintValue = reader.readSINT();

            //  "MAIN.intValue" // INT
            var intValue = reader.readINT();

            //  "MAIN.dintValue" // DINT
            var dintValue = reader.readDINT();

            //  "MAIN.realValue" // REAL
            var realValue = reader.readREAL();

            //  "MAIN.lrealValue" // LREAL
            var lrealValue = reader.readLREAL();

            // Write data to the user interface;
            td_byteValue.innerHTML = byteValue;
            td_wordValue.innerHTML = wordValue;
            td_dwordValue.innerHTML = dwordValue;
            td_sintValue.innerHTML = sintValue;
            td_intValue.innerHTML = intValue;
            td_dintValue.innerHTML = dintValue;
            td_realValue.innerHTML = realValue;
            td_lrealValue.innerHTML = lrealValue;

        } else {

            if (e.error.getTypeString() == "TcAdsWebService.ResquestError") {
            // HANDLE TcAdsWebService.ResquestError HERE;
            div_log.innerHTML = "Error: StatusText = " + e.error.statusText + " Status: " + e.
error.status;
            }
            else if (e.error.getTypeString() == "TcAdsWebService.Error") {
            // HANDLE TcAdsWebService.Error HERE;
            div_log.innerHTML = "Error: ErrorMessage = " + e.error.errorMessage + " ErrorCode:
 " + e.error.errorCode;
            }
        }

        });

        // Occurs if the read-read-write command runs into timeout;
        var ReadTimeoutCallback = (function () {
        // HANDLE TIMEOUT HERE;
        div_log.innerHTML = "Read timeout!";
        });

    })();
    </script>
</head>
<body>
```

```
    <h1>TcAdsWebService.js Sample01</h1>
    <h2>Cyclic reading of multiple variables with sumcommando!</h2>
    <table>
    <tr>
        <td>MAIN.byteValue:</td>
        <td id="td_byteValue"></td>
    </tr>
    <tr>
        <td>MAIN.wordValue:</td>
        <td id="td_wordValue"></td>
    </tr>
    <tr>
        <td>MAIN.dwordValue:</td>
        <td id="td_dwordValue"></td>
    </tr>
    <tr>
        <td>MAIN.sintValue:</td>
        <td id="td_sintValue"></td>
    </tr>
    <tr>
        <td>MAIN.intValue:</td>
        <td id="td_intValue"></td>
    </tr>
    <tr>
        <td>MAIN.dintValue:</td>
        <td id="td_dintValue"></td>
    </tr>
    <tr>
        <td>MAIN.realValue:</td>
        <td id="td_realValue"></td>
    </tr>
    <tr>
        <td>MAIN.lrealValue:</td>
        <td id="td_lrealValue"></td>
    </tr>
    </table>
    <div id="div_log"></div>
</body>
</html>
```

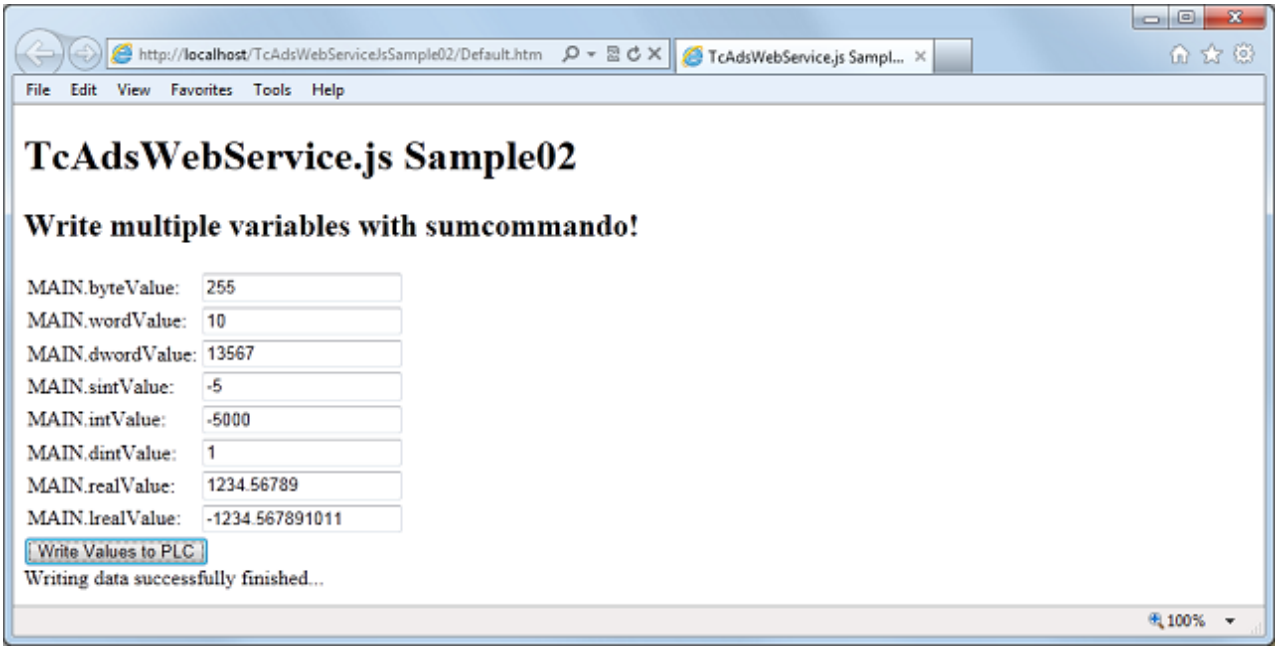
6.2.2 Writing multiple variables with sumcommando

Description

This sample shows how to write multiple variables by use of a write sumcommando.

Transporting bigger data packets

Generally, HTTP-traffic is more efficient by transporting bigger data packets instead of many small packets. Therefore, please use ADS sum commandos to collect your data.



ToDo:

- Download the https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488971659/.zip
- Start the PLC project
- Copy the default.htm and TcAdsWebService.js to your cofigured virtual directory
- Adjust NETID, PORT and SERVICE_URL if you use the TcAdsWebservice of a remote pc

```
<!DOCTYPE html>
<html>
<head>
    <title>TcAdsWebService.js Sample01</title>
    <script type="text/javascript" src="TcAdsWebService.js"></script>
    <script type="text/javascript">
    (function () {

        var NETID = ""; // Empty string for local machine;
        var PORT = "801"; // TC2 PLC Runtime = 801, TC3 PLC Runtime = 851
        var SERVICE_URL = "http://localhost/TcAdsWebService/
TcAdsWebService.dll"; // HTTP path to the TcAdsWebService;

        var client = new TcAdsWebService.Client(SERVICE_URL, null, null);

        var general_timeout = 500;

        // Array of symbol names to read;
        var handlesVarNames = [
            "MAIN.byteValue",
            "MAIN.wordValue",
            "MAIN.dwordValue",
            "MAIN.sintValue",
            "MAIN.intValue",
            "MAIN.dintValue",
            "MAIN.realValue",
            "MAIN.lrealValue"
        ];

        var handlesVarSizes = [ 1, 2, 4, 1, 2, 4, 4, 8 ];

        var handles = [];


        // Occurs if the window has loaded;
        window.onload = (function () {

        // Register ui events
```

```
        btnWrite.onclick = BtnWriteClicked;

        // Create sumcommando for reading twincat symbol handles by symbol name;
        var handleswriter = new TcAdsWebService.DataWriter();

        // Write general information for each symbol handle to the TcAdsWebService.DataWriter object
;
        for (var i = 0; i < handlesVarNames.length; i++) {
            handleswriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolHandleByName);
            handleswriter.writeDINT(0);
            handleswriter.writeDINT(4); // Expected size; A handle has a size of 4 byte;
            handleswriter.writeDINT(handlesVarNames[i].length); // The length of the symbol name str
ing;
        }

        // Write symbol names after the general information to the TcAdsWebService.DataWriter object
;
        for (var i = 0; i < handlesVarNames.length; i++) {
            handleswriter.writeString(handlesVarNames[i]);
        }

        // Send the list-read-
write command to the TcAdsWebService by use of the readwrite function of the TcAdsWebService.Client
object;
        client.readwrite(
            NETID,
            PORT,
            0xF082,     // IndexGroup = ADS list-read-
write command; Used to request handles for twincat symbols;
            handlesVarNames.length, // IndexOffset = Count of requested symbol handles;
            (handlesVarNames.length * 4) + (handlesVarNames.length * 8), // Length of requested data
 + 4 byte errorcode and 4 byte length per twincat symbol;
            handleswriter.getBase64EncodedData(),
            RequestHandlesCallback,
            null,
            general_timeout,
            RequestHandlesTimeoutCallback,
            true);

        });

        // Occurs if the readwrite for the sumcommando has finished;
        var RequestHandlesCallback = (function (e, s) {

        if (e && e.isBusy) {
            // HANDLE PROGRESS TASKS HERE;
            var message = "Requesting symbol handles...";
            div_log.innerHTML = message;
            // Exit callback function because request is still busy;
            return;
        }

        if (e && !e.hasError) {

            // Get TcAdsWebService.DataReader object from TcAdsWebService.Response object;
            var reader = e.reader;

            // Read error code and length for each handle;
            for (var i = 0; i < handlesVarNames.length; i++) {

            var err = reader.readDWORD();
            var len = reader.readDWORD();

            if (err != 0) {
                div_log.innerHTML = "Symbol handle error!";
                return;
            }

            }

            var message = "Symbol handles successfully created!";
            div_log.innerHTML = message;

            // Read handles from TcAdsWebService.DataReader object;
            handles[0] = reader.readDWORD();
            handles[1] = reader.readDWORD();
            handles[2] = reader.readDWORD();
            handles[3] = reader.readDWORD();
            handles[4] = reader.readDWORD();
            handles[5] = reader.readDWORD();
```

```
            handles[6] = reader.readDWORD();
            handles[7] = reader.readDWORD();

        } else {

            if (e.error.getTypeString() == "TcAdsWebService.ResquestError") {
            // HANDLE TcAdsWebService.ResquestError HERE;
            div_log.innerHTML = "Error: StatusText = " + e.error.statusText + " Status: " + e.error.
status;
            }
            else if (e.error.getTypeString() == "TcAdsWebService.Error") {
            // HANDLE TcAdsWebService.Error HERE;
            div_log.innerHTML = "Error: ErrorMessage = " + e.error.errorMessage + " ErrorCode: " + e
.error.errorCode;
            }

        }

        });

        // Occurs if the readwrite for the sumcommando to request symbol handles runs into timeout;
        var RequestHandlesTimeoutCallback = (function () {
        // HANDLE TIMEOUT HERE;
        div_log.innerHTML = "Requuest symbol handles timeout!";
        });

          var BtnWriteClicked = (function () {

        // Create TcAdsWebService.DataWriter for write-read-write command.
        var writer = new TcAdsWebService.DataWriter();

        // Write general write-read-write commando information to TcAdsWebService.DataWriter object;
        var size = 0;
        for (var i = 0; i < handles.length; i++) {
            writer.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolValueByHandle);
            writer.writeDINT(handles[i]);
            writer.writeDINT(handlesVarSizes[i]);

            size = size + handlesVarSizes[i];
        }

        // Write values to TcAdsWebService.DataWrite object;
        writer.writeBYTE(parseFloat(Text1.value));
        writer.writeWORD(parseFloat(Text2.value));
        writer.writeDWORD(parseFloat(Text3.value));
        writer.writeSINT(parseFloat(Text4.value));
        writer.writeINT(parseFloat(Text5.value));
        writer.writeDINT(parseFloat(Text6.value));
        writer.writeREAL(parseFloat(Text7.value));
        writer.writeLREAL(parseFloat(Text8.value));

        client.readwrite(
            NETID,
            PORT,
            0xF081, // 0xF081 = Call Write SumCommando
            handles.length, // IndexOffset = Count of requested variables.
            size+(handles.length*4), // Length of requested data + 4 byte errorcode per variable.
            writer.getBase64EncodedData(),
            WriteCallback,
            null,
            general_timeout,
            WriteTimeoutCallback,
            true);
        });

        var WriteCallback = (function(e,s){

        if (e && e.isBusy) {
            // HANDLE PROGRESS TASKS HERE;
            var message = "Writing data to plc...";
            div_log.innerHTML = message;
            // Exit callback function because request is still busy;
            return;
        }

        if (e && !e.hasError) {

            var message = "Writing data successfully finished...";
            div_log.innerHTML = message;
```

```
        } else {

            if (e.error.getTypeString() == "TcAdsWebService.ResquestError") {
            // HANDLE TcAdsWebService.ResquestError HERE;
            div_log.innerHTML = "Error: StatusText = " + e.error.statusText + " Status: " + e.error.
status;
            }
            else if (e.error.getTypeString() == "TcAdsWebService.Error") {
            // HANDLE TcAdsWebService.Error HERE;
            div_log.innerHTML = "Error: ErrorMessage = " + e.error.errorMessage + " ErrorCode: " + e
.error.errorCode;
            }

        }
        });

        // Occurs if the write-read-write command runs into timeout;
        var WriteTimeoutCallback = (function () {
        // HANDLE TIMEOUT HERE;
        div_log.innerHTML = "Write timeout!";
        });

    })();
    </script>
</head>
<body>
    <h1>TcAdsWebService.js Sample02</h1>
    <h2>Write multiple variables with sumcommando!</h2>
    <table>
    <tr>
        <td>MAIN.byteValue:</td>
        <td><input id="Text1" type="text" /></td>
    </tr>
    <tr>
        <td>MAIN.wordValue:</td>
        <td><input id="Text2" type="text" /></td>
    </tr>
    <tr>
        <td>MAIN.dwordValue:</td>
        <td><input id="Text3" type="text" /></td>
    </tr>
    <tr>
        <td>MAIN.sintValue:</td>
        <td><input id="Text4" type="text" /></td>
    </tr>
    <tr>
        <td>MAIN.intValue:</td>
        <td><input id="Text5" type="text" /></td>
    </tr>
    <tr>
        <td>MAIN.dintValue:</td>
        <td><input id="Text6" type="text" /></td>
    </tr>
    <tr>
        <td>MAIN.realValue:</td>
        <td><input id="Text7" type="text" /></td>
    </tr>
    <tr>
        <td>MAIN.lrealValue:</td>
        <td><input id="Text8" type="text" /></td>
    </tr>
    </table>
    <input type="button" id="btnWrite" value="Write Values to PLC" />
    <div id="div_log"></div>
</body>
</html>
```

6.2.3 Example: Maschine.pro with HTML5, SVG, JavaScript

Web based standard technologies like HTML5, SVG (Scaleable Vector Graphics) and JavaScript are perfect for creating user interfaces which are platform independend and easy to deploy.

Required software:

- HTML Editor
- TwinCAT 2.11

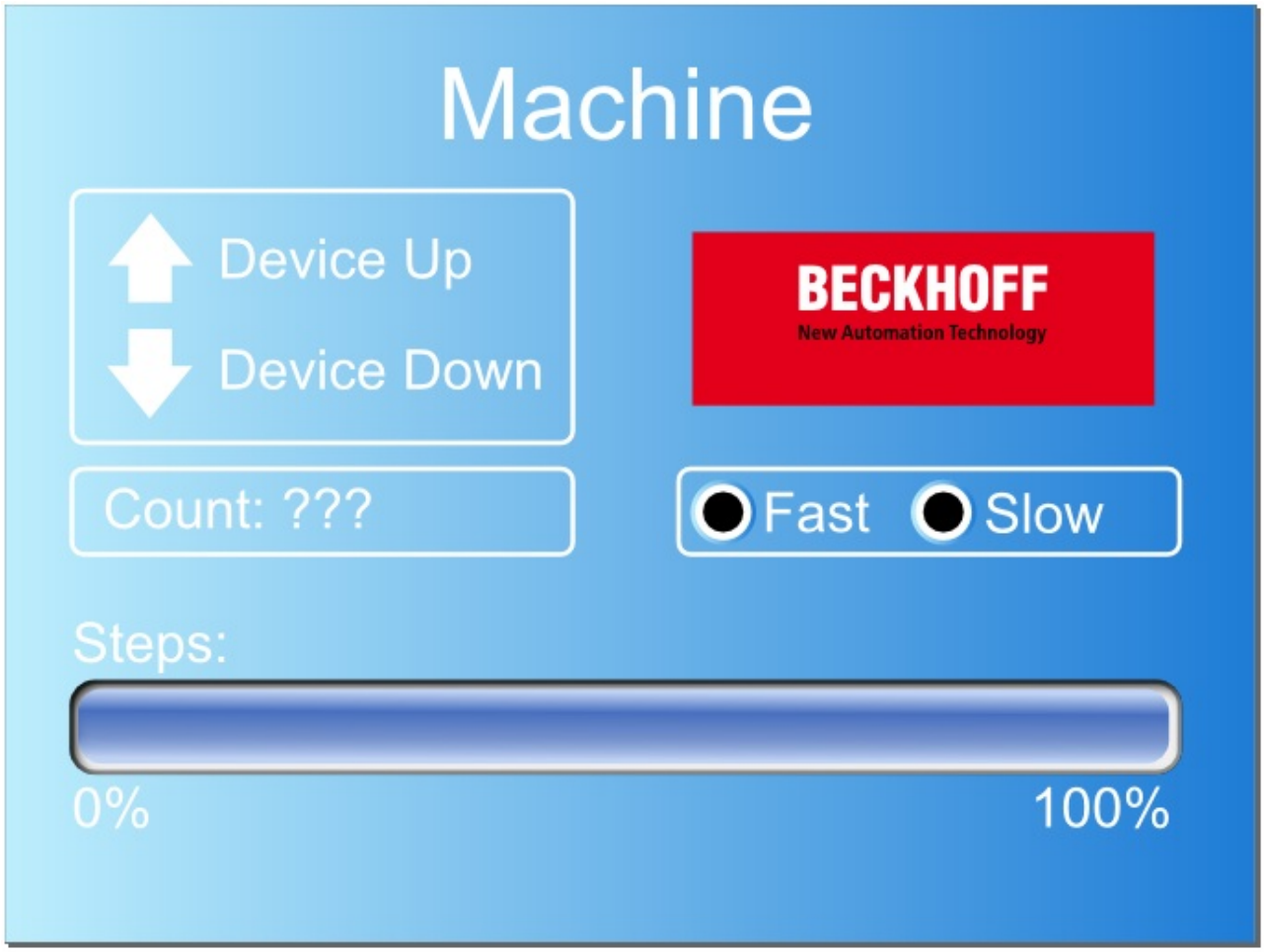
- TcAdsWebService
- TcAdsWebService.js Library
- Microsoft IIS (Internet Information Services) Webserver

Before you can start...

- TwinCAT and the PLC program must be active.
- Microsoft IIS has to be installed and configured for hosting the TcAdsWebService.

1. Creating the SVG for the user interface

Use a vector based design software of your choice to create the draft for the SVG user interface and make sure that all elements which you want to manipulate from JavaScript are named reasonable.



2. Creating the basic HTML5 document

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=9" />
    <title>HTML5. SVG, JavaScript Machine Sample</title>
    <link rel="Stylesheet" href=StyleSheet.css />
    <script type="text/javascript" src="TcAdsWebService.js"></script>
    <script type="text/javascript">
    <!-- JavaScript will be placed here -->
    </script>
</head>
<body>
<!-- SVG will be placed here -->
</body>
</html>
```

This meta tag enables inline svg support for the Internet Explorer 9.

```
<meta http-equiv="X-UA-Compatible" content="IE=9" />
```

2. Implement the SVG code into the body of the HTML5 document

```
{...}
<body>
  <svg xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns="http://www.w3.org/2000/
svg" xmlns:cc="http://creativecommons.org/ns#"
       xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:dc="http://purl.org/dc/elements/1.1/"
       id="svgROOT" height="100%" width="100%" viewBox="0 0 800 600" version="1.1" style="display: b
lock;">
    {...}
   </svg>
</body>
{...}
```

To grant the optimal displaying of the SVG user interface we have to add some additional information to the SVG root element.

- The id is set to "svgROOT". This identifier can be used in JavaScript to get access to the SVG elements.
- The "height" and "width" properties are set to 100% because we want the user interface to use all the available space in the browser.
- The "viewBox" property defines the position and the original size of the SVG in the pattern "x-coordinate y-coordinate width height".
  Because of this, the SVG will only scale within its aspect ratio.

2. Implement the JavaScript based logic

Our code is created within a function which is called directly by the JavaScript interpreter to prevent from conflicts with other JavaScript elements which may exist later in this HTML document.

```
(function (window) {
    // We create our code here;
})(window);
```

At fist we define variables which exists in the whole context of our directly called JavaScript function.

```
var NETID = ""; // Empty string for local machine;
var PORT = "801"; // PLC Runtime port;
var SERVICE_URL = "http://" + window.document.location.hostname + "/TcAdsWebService/
TcAdsWebService.dll"; // HTTP path to the target TcAdsWebService;

// The TcAdsWebService.Client object for ajax communication with the TcAdsWebService;
var client = new TcAdsWebService.Client(SERVICE_URL, null, null);

var general_timeout = 500; // This timeout value is used for all requests;

var readLoopID = null; // The id of the read interval; Can be used to stop the polling if need;
var readLoopDelay = 100;

var readSymbolValuesData = null; // This variable will store the Base64 encoded binary data string f
or the read sumcommando;

// Array of symbol names to read;
var handlesVarNames = [
    ".engine",
    ".deviceUp",
    ".deviceDown",
    ".steps",
    ".count",
    ".switch"
    ];

// Symbol handle variables;
```

```
var hEngine = null;
var hDeviceUp = null;
var hDeviceDown = null;
var hSteps = null;
var hCount = null;
var hSwitch = null;

// Base64 encoded binary data strings for write requests;
var switchTrueBase64 = null;
var switchFalseBase64 = null;

// UI Elements;
var radioActiveFast = null;
var radioActiveSlow = null;
var radioBackgroundFast = null;
var radioBackgroundSlow = null;
var textCount = null;
var upArrow = null;
var downArrow = null;
var progress = null;
var progressBackground = null;

var progress_initial_width = 0;
```

The initialization is done in the callback function of the window.onload event.
We do this, because we can be sure that all SVG and HTML elements do exist now.

```
window.onload = (function () { /* Initialization is done here */ });
```

Initializing

```
 // Initialize UI Elements;
radioActiveFast = svgROOT.getElementById("radioActiveFast");
radioActiveSlow = svgROOT.getElementById("radioActiveSlow");
radioBackgroundFast = svgROOT.getElementById("radioBackgroundFast");
radioBackgroundSlow = svgROOT.getElementById("radioBackgroundSlow");
textCount = svgROOT.getElementById("textCount");
upArrow = svgROOT.getElementById("UPArrow");
downArrow = svgROOT.getElementById("DownArrow");
progress = svgROOT.getElementById("progress");

progress_initial_width = progress.width.baseVal.value;

radioBackgroundFast.onclick = RadioBackgroundFastClick;
radioBackgroundSlow.onclick = RadioBackgroundSlowClick;

// Prepare data for writing to switch variable;
var switchTrueBase64Writer = new TcAdsWebService.DataWriter();
switchTrueBase64Writer.writeBOOL(true);
switchTrueBase64 = switchTrueBase64Writer.getBase64EncodedData();

var switchFalseBase64Writer = new TcAdsWebService.DataWriter();
switchFalseBase64Writer.writeBOOL(false);
switchFalseBase64 = switchFalseBase64Writer.getBase64EncodedData();
```

Now we initialize a TcAdsWebService.DataWriter object to create a Base64 encoded binary string which contains all the data which the sumcommando needs to request handles for the TwinCAT Symbol names in the "handlesVarNames" array.
With the "readwrite" function of the TcAdsWebService.Client object we start the sumcommando request against the TcAdsWebService.

```
 // Create sumcommando for reading twincat symbol handles by symbol name;
var handleswriter = new TcAdsWebService.DataWriter();

// Write general information for each symbol handle to the TcAdsWebService.DataWriter object;
for (var i = 0; i < handlesVarNames.length; i++) {
    handleswriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolHandleByName);
    handleswriter.writeDINT(0);
    handleswriter.writeDINT(4); // Expected size; A handle has a size of 4 byte;
    handleswriter.writeDINT(handlesVarNames[i].length); // The length of the symbol name string;
}
```

```
// Write symbol names after the general information to the TcAdsWebService.DataWriter object;
for (var i = 0; i < handlesVarNames.length; i++) {
    handleswriter.writeString(handlesVarNames[i]);
}

// Send the list-read-
write command to the TcAdsWebService by use of the readwrite function of the TcAdsWebService.Client
object;
client.readwrite(
    NETID,
    PORT,
    0xF082, // IndexGroup = ADS list-read-
write command; Used to request handles for twincat symbols;
    handlesVarNames.length, // IndexOffset = Count of requested symbol handles;
    (handlesVarNames.length * 4) + (handlesVarNames.length * 8), // Length of requested data + 4 byt
e errorcode and 4 byte length per twincat symbol;
    handleswriter.getBase64EncodedData(),
    RequestHandlesCallback,
    null,
    general_timeout,
    RequestHandlesTimeoutCallback,
    true);
```

If the sumcommando to request the TwinCAT Symbol handles has finished, the RequestHandlesCallback function is called.
The parameter "e" will contain a TcAdsWebService.Response object with information about the request against the TcAdsWebService.
The parameter "s" is the "userState" parameter which was set to "null" in the function call.

```
// Occurs if the readwrite for the sumcommando has finished;
var RequestHandlesCallback = (function (e, s) {
    {...}
});
```

Within the RequestHandlesCallback function we collect the handle information from the TcAdsWebService.DataReader object in the "reader" property of the TcAdsWebService.Response object. With these handles we initialize another TcAdsWebService.DataWriter object with data for a read sumcommando which will be called cyclic in the ReadLoop function.

```
 if (e && e.isBusy) {
    // HANDLE PROGRESS TASKS HERE;
    // Exit callback function because request is still busy;
    return;
}

if (e && !e.hasError) {

    // Get TcAdsWebService.DataReader object from TcAdsWebService.Response object;
    var reader = e.reader;

    // Read error code and length for each handle;
    for (var i = 0; i < handlesVarNames.length; i++) {

    var err = reader.readDWORD();
    var len = reader.readDWORD();

    if (err != 0) {
        // HANDLE SUMCOMMANDO ERRORS HERE;
        return;
    }

    }

    // Read handles from TcAdsWebService.DataReader object;
    hEngine = reader.readDWORD();
    hDeviceUp = reader.readDWORD();
    hDeviceDown = reader.readDWORD();
    hSteps = reader.readDWORD();
    hCount = reader.readDWORD();
    hSwitch = reader.readDWORD();

    // Create sum commando to read symbol values based on the handle;
    var readSymbolValuesWriter = new TcAdsWebService.DataWriter();
```

```
    //  ".engine" // BOOL
    readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolValueByHandle);
// IndexGroup
    readSymbolValuesWriter.writeDINT(hEngine); // IndexOffset = The target handle
    readSymbolValuesWriter.writeDINT(1); // Size to read;

    //  ".deviceUp" // BOOL
    readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolValueByHandle);
// IndexGroup
    readSymbolValuesWriter.writeDINT(hDeviceUp); // IndexOffset = The target handle
    readSymbolValuesWriter.writeDINT(1); // Size to read;

    //  ".deviceDown" // BOOL
    readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolValueByHandle);
// IndexGroup
    readSymbolValuesWriter.writeDINT(hDeviceDown); // IndexOffset = The target handle
    readSymbolValuesWriter.writeDINT(1); // Size to read;

    //  ".steps" // BYTE
    readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolValueByHandle);
// IndexGroup
    readSymbolValuesWriter.writeDINT(hSteps); // IndexOffset = The target handle
    readSymbolValuesWriter.writeDINT(1); // Size to read;

    //  ".count" // UINT
    readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolValueByHandle);
// IndexGroup
    readSymbolValuesWriter.writeDINT(hCount); // IndexOffset = The target handle
    readSymbolValuesWriter.writeDINT(2); // Size to read;

    //  ".switch" // BOOL
    readSymbolValuesWriter.writeDINT(TcAdsWebService.TcAdsReservedIndexGroups.SymbolValueByHandle);
// IndexGroup
    readSymbolValuesWriter.writeDINT(hSwitch); // IndexOffset = The target handle
    readSymbolValuesWriter.writeDINT(1); // Size to read;

    // Get Base64 encoded data from TcAdsWebService.DataWriter;
    readSymbolValuesData = readSymbolValuesWriter.getBase64EncodedData();

    // Start cyclic reading of symbol values;
    readLoopID = window.setInterval(ReadLoop, readLoopDelay);

    } else {

    if (e.error.getTypeString() == "TcAdsWebService.ResquestError") {
    // HANDLE TcAdsWebService.ResquestError HERE;
    }
    else if (e.error.getTypeString() == "TcAdsWebService.Error") {
    // HANDLE TcAdsWebService.Error HERE;
    }

}
```

The ReadLoop function calls the the read sumcommando on every tick;

```
// Interval callback for cyclic reading;
var ReadLoop = (function () {

    // Send the read-read-
write command to the TcAdsWebService by use of the readwrite function of the TcAdsWebService.Client
object;
    client.readwrite(
    NETID,
    PORT,
    0xF080, // 0xF080 = Read command;
    handlesVarNames.length, // IndexOffset = Variables count;
    7 + (handlesVarNames.length * 4), // Length of requested data + 4 byte errorcode per variable;
    readSymbolValuesData,
    ReadCallback,
    null,
    general_timeout,
    ReadTimeoutCallback,
    true);

});
```

In the ReadCallback function we collect the values of each TwinCAT Symbol in the TcAdsWebService.DataReader object of the "reader" porperty of the TcAdsWebService.Response object and use them to update the user interface.

```
// Occurs if the read-read-write command has finished;
var ReadCallback = (function (e, s) {

    if (e && e.isBusy) {
    // HANDLE PROGRESS TASKS HERE;
    // Exit callback function because request is still busy;
    return;
    }

    if (e && !e.hasError) {

    var reader = e.reader;

    // Read error codes from begin of TcAdsWebService.DataReader object;
    for (var i = 0; i < handlesVarNames.length; i++) {
        var err = reader.readDWORD();
        if (err != 0) {
        // HANDLE SUMCOMMANDO ERRORS HERE;
        return;
        }
    }

    // READ Symbol data from TcAdsWebService.DataReader object;
    //  ".engine" // BOOL
    var engine = reader.readBOOL();
    //  ".deviceUp" // BOOL
    var deviceUp = reader.readBOOL();
    //  ".deviceDown" // BOOL
    var deviceDown = reader.readBOOL();
    //  ".steps" // BYTE
    var steps = reader.readBYTE();
    //  ".count" // UINT/WORD
    var count = reader.readWORD();
    //  ".switch" // BOOL
    var switchVar = reader.readBYTE();

    // Update UI
    // Radio buttons
    if (switchVar) {
        radioActiveFast.style.visibility = "visible";
        radioActiveSlow.style.visibility = "hidden";
    } else {
        radioActiveFast.style.visibility = "hidden";
        radioActiveSlow.style.visibility = "visible";
    }

    // Textblock: Count
    textCount.textContent = "Count: " + count;

    // DeviceUp / DeviceDown Arrows
    if (deviceUp) {
        upArrow.style.fill = "#ea0000";
    } else {
        upArrow.style.fill = "#ffffff";
    }

    if (deviceDown) {
        downArrow.style.fill = "#ea0000";
    } else {
        downArrow.style.fill = "#ffffff";
    }

    // Progress
    var progress_scale = steps / 25; // 25 = max value of steps symbol;
    if (steps >= 0 && steps <= 25) {
        progress.width.baseVal.value = progress_initial_width * progress_scale;
    }
    else if (steps < 0) {
        progress.width.baseVal.value = 0.00;
    }
    else if (steps > 25) {
        progress.width.baseVal.value = progress_initial_width;
    }
```

```
    } else {

    if (e.error.getTypeString() == "TcAdsWebService.ResquestError") {
        // HANDLE TcAdsWebService.ResquestError HERE;
    }
    else if (e.error.getTypeString() == "TcAdsWebService.Error") {
        // HANDLE TcAdsWebService.Error HERE;
    }
    }

});
```

If the RadioBackgroundFastClick function or the RadioBackgroundSlowClick functions are called we send a "write" request with the specified Base64 encoded binary data against the TcAdsWebService.

```
// Occurs if the background of the radio button "Fast" is clicked;
var RadioBackgroundFastClick = (function () {

    client.write(
    NETID,
    PORT,
    0x0000F005, // IndexGroup = Write variable by handle;
    hSwitch,
    switchTrueBase64,
    RadioBackgroundFastWriteCallback,
    null,
    general_timeout,
    RadioBackgroundFastWriteTimeoutCallback,
    true
    );

});
```

```
// Occurs if the background of the radio button "Slow" is clicked;
var RadioBackgroundSlowClick = (function () {

    client.write(
    NETID,
    PORT,
    0x0000F005, // Write variable by handle;
    hSwitch,
    switchFalseBase64,
    RadioBackgroundSlowWriteCallback,
    null,
    general_timeout,
    RadioBackgroundSlowWriteTimeoutCallback,
    true
    );

});
```

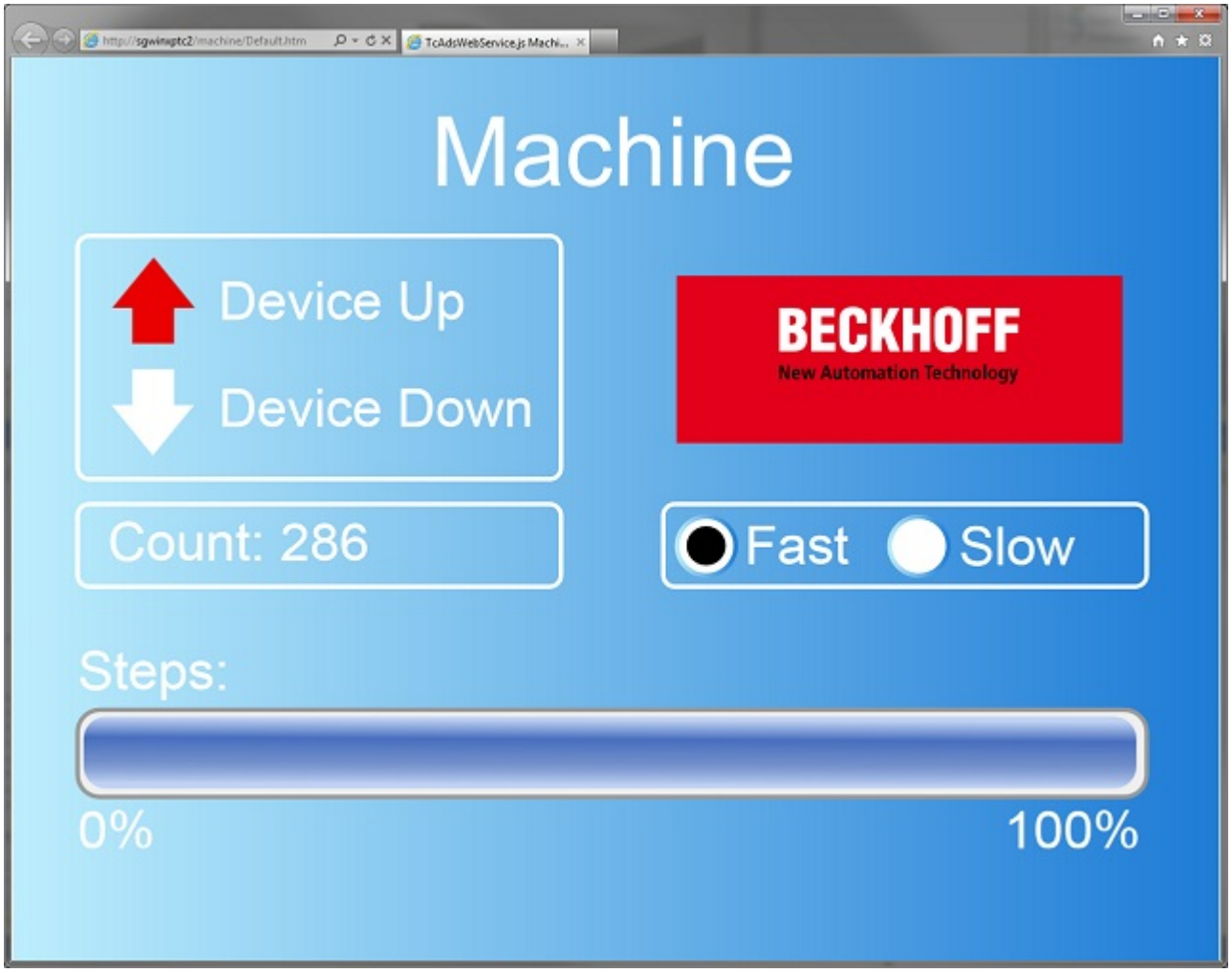
If the browser window or the browser tab is going to be closed we release the used handles with the IndexGroup 0xF006 (Relase Symbol Handle).

```
window.onbeforeunload = (function () {

    // Free Handles
    client.write(NETID, PORT, 0xF006, hEngine, "", FreeHandleCallback, "hEngine", general_timeout, F
reeHandleTimeoutCallback, true);
    client.write(NETID, PORT, 0xF006, hDeviceUp, "", FreeHandleCallback, "hDeviceUp", general_timeou
t, FreeHandleTimeoutCallback, true);
    client.write(NETID, PORT, 0xF006, hDeviceDown, "", FreeHandleCallback, "hDeviceDown", general_ti
meout, FreeHandleTimeoutCallback, true);
    client.write(NETID, PORT, 0xF006, hSteps, "", FreeHandleCallback, "hSteps", general_timeout, Fre
eHandleTimeoutCallback, true);
    client.write(NETID, PORT, 0xF006, hCount, "", FreeHandleCallback, "hCount", general_timeout, Fre
eHandleTimeoutCallback, true);
    client.write(NETID, PORT, 0xF006, hSwitch, "", FreeHandleCallback, "hSwitch", general_timeout, F
reeHandleTimeoutCallback, true);

});
```

Now we can watch the Machine Sample in any Browser with HTML5 and inline SVG support.



2. Download the example

https://infosys.beckhoff.com/content/1033/tcadswebservice/Resources/12488973067/.zip

More Information:
www.beckhoff.com/automation

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com