BECKHOFF New Automation Technology

Manual | EN

TX1000

TwinCAT 2 | ADS WCF

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

Trademarks

Patent Pending

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.

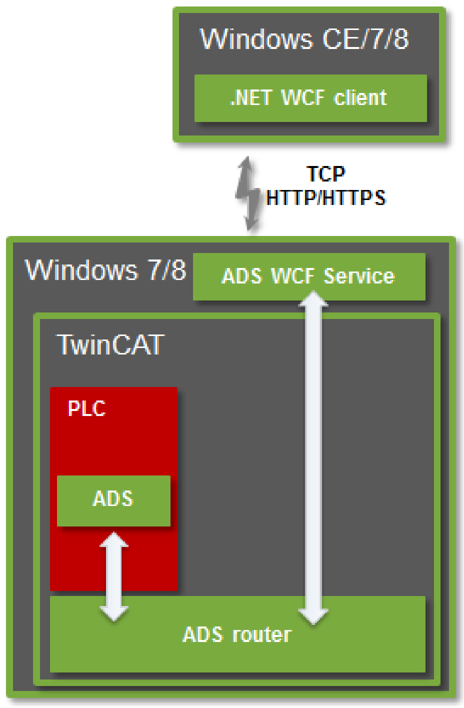
In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

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

The TwinCAT ADS WCF Service is based on the TwinCAT.Ads DLL and provides access to ads devices via Windows Communication Foundation (WCF) technology.



- WCF (**W**indows **C**ommunication **F**oundation)
  - Part of the .NET framework from Microsoft
  - Used to build service-oriented applications (SOA)
- Rapidly implemented
  - WCF clients does not require a TwinCAT installation
  - ADS route configuration is not needed
- Security
  - Provide SSL encryption
  - Support authentication
- Platforms
  - Support of x86-, x64-platforms
  - CE support for WCF clients
  - .NET Framework is required

API
Simplex Methods [▸ 14]
Duplex Methods [▸ 28]
Callbacks [▸ 52]
Endpoints [▸ 66]

3 Installation

3.1 Requirements

Server

Supported operating system: Windows XP, XP Embedded, Embedded Standard 2009, Windows 7, Windows 8

TwinCAT Level: minimum TwinCAT2 CP or TwinCAT3 ADS

.NET Framework: minimum v2.0.50727

Supported by **Windows CE: no**

Client

Supported operating system: Windows XP, XP Embedded, Embedded Standard 2009, Windows 7, Windows 8

TwinCAT Level: no TwinCAT Installation is required

.NET Framework: minimum v2.0.50727

Supported by **Windows CE: Yes**

.NET Compact Framework: v3.5 (for CE)

3.2 Service Installation

The following step descripes how to install the TwinCAT ADS WCF Service, which can be connected by a ADS WCF client.

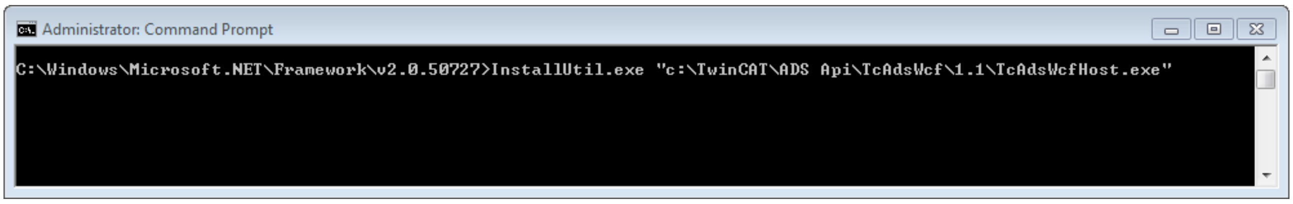
1. Install libraries

Install the listed libraries into the "Global Assembly Cache" by copying them to "C:\WINDOWS\assembly" as Administrator:

- "TwinCAT\AdsApi\TcAdsWcf\PollingDuplex\v3.0\System.ServiceModel.PollingDuplex.dll"
- "TwinCAT\AdsApi\TcAdsWcf\PollingDuplex\v4.0\System.ServiceModel.PollingDuplex.dll"

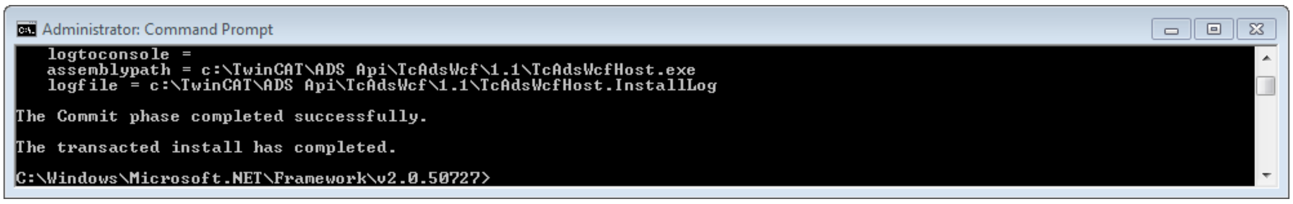
2. Create Windows Service

Open the command prompt as Administrator and use the InstallUtil.exe (part of the .NET Framework) to create the Windows service:



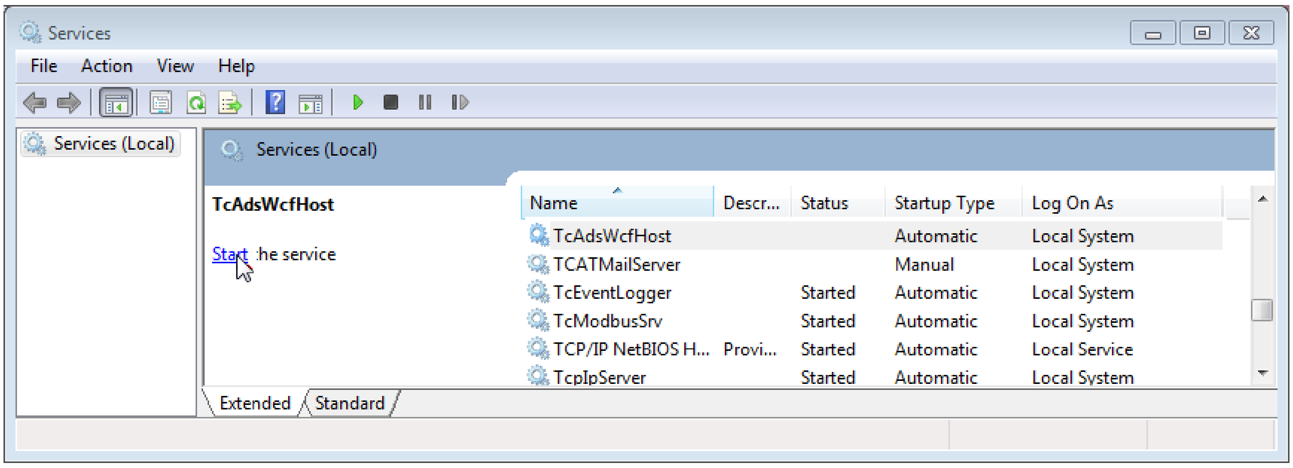
```
C:\Windows\Microsoft.NET\Framework\v2.0.50727>InstallUtil.exe "c:\TwinCAT\ADS Api\TcAdsWcf\1.1\TcAdsWcfHost.exe"
```

You will be informed, if the service is installed ordinary:



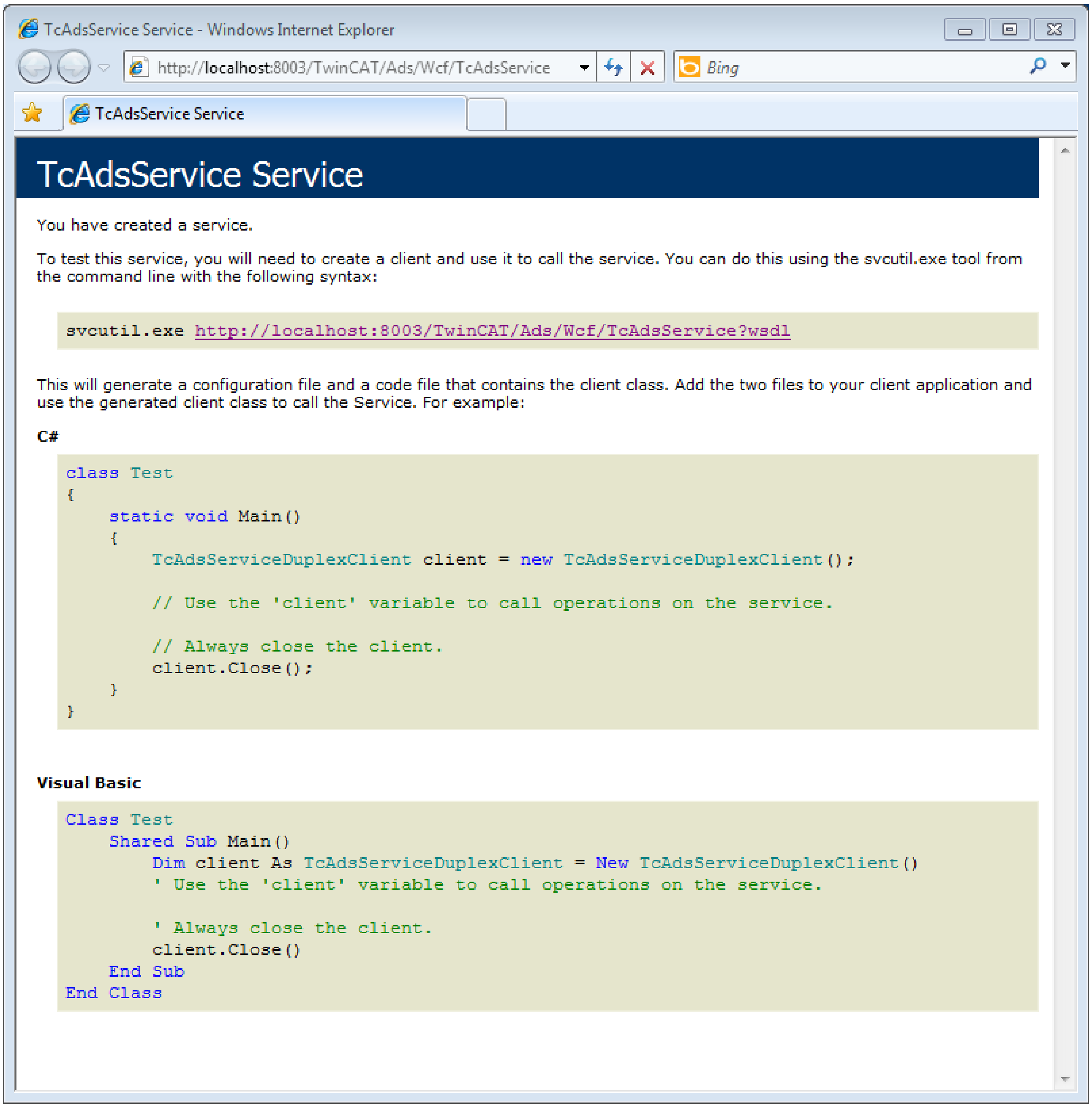
```
   logtoconsole =
   assemblypath = c:\TwinCAT\ADS Api\TcAdsWcf\1.1\TcAdsWcfHost.exe
   logfile = c:\TwinCAT\ADS Api\TcAdsWcf\1.1\TcAdsWcfHost.InstallLog

The Commit phase completed successfully.

The transacted install has completed.

C:\Windows\Microsoft.NET\Framework\v2.0.50727>
```

Switch to "Control Panel\Administrative Tools\Services" and start the "TcAdsWcfHost" Service:



Name	Descr...	Status	Startup Type	Log On As
TcAdsWcfHost			Automatic	Local System
TCATMailServer			Manual	Local System
TcEventLogger		Started	Automatic	Local System
TcModbusSrv		Started	Automatic	Local System
TCP/IP NetBIOS H...	Provi...	Started	Automatic	Local Service
TcpIpServer		Started	Automatic	Local System

Open a browser and go to *http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService* to test the service. You will receive following webpages:


TcAdsService Service - Windows Internet Explorer
http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService
Bing
TcAdsService Service

TcAdsService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService?wsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

```
class Test
{
    static void Main()
    {
        TcAdsServiceDuplexClient client = new TcAdsServiceDuplexClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Visual Basic

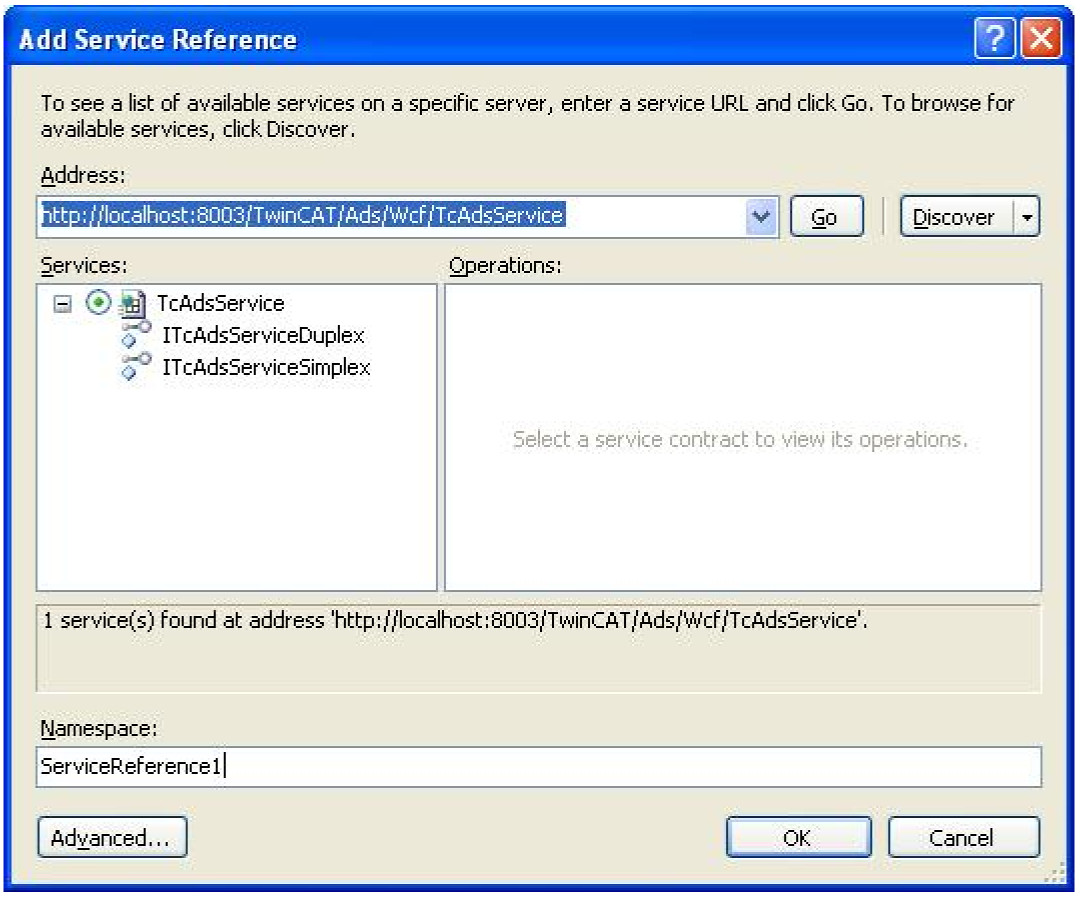
```
Class Test
    Shared Sub Main()
        Dim client As TcAdsServiceDuplexClient = New TcAdsServiceDuplexClient()
        ' Use the 'client' variable to call operations on the service.

        ' Always close the client.
        client.Close()
    End Sub
End Class
```

Sample Overview [▸ 70]

If you want to uninstall the TwinCAT ADS WCF Service use the parameter -u of the InstallUtil.exe-Tool (https://docs.microsoft.com/en-us/dotnet/framework/tools/installutil-exe-installer-tool).

3.3 Create a WCF service reference in Visual Studio

In order to create a WCF service reference in Visual Studio you must choose the command *Add Service Reference...* under the *Project* menu.
This opens the Add Service Reference dialog:



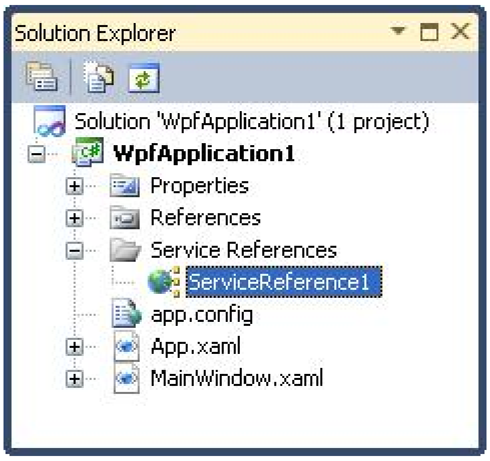
To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService

1 service(s) found at address 'http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService'.

Namespace: ServiceReference1

In the dialog enter the http or the https based base address of the TcAdsService and press the *GO* button. This will load the service meta data and create the needed client code.
After that the service reference will be listed in the Solution Explorer with the chosen namespace.



4 Security references

The TwinCAT ADS WCF service provides access through different endpoints [▶ 66]. Some of them are implemented without security features to provide better performance and should only be used in secured intranet environments. These endpoints [▶ 66] are marked with the *Unsec* prefix.

If you want the TwinCAT ADS WCF service to be reachable via the Internet, you should ensure that these endpoints [▶ 66] are not reachable.

Secured endpoints

If you want to use the secured endpoints of the TwinCAT ADS WCF service you have to use a SSL encoded port for communication and the authentication is based on Windows credentials.

You can use your own certificates for SSL encoding or you can use a self signed certificate generated by the TcAdsWcfCertGen.exe [▶ 13] tool.

Configure a port with an SSL certificate

Microsoft provides port configuration tools for the different versions of Microsoft Windows. A manual for configuring a port with an SSL certificate can be found here.

4.1 TcAdsWcfCertGen.exe

TcAdsWcfCertGen is a console application which will create a self signed certificate for SSL encoding. TcAdsWcfCertGen is stored in the TcAdsWcf installation directory.

Usage: TcAdsWcfCertGen.exe DOMAINNAME

Parameters

DOMAINNAME

The name of the Domain which will be used to reach the TwinCAT ADS WCF service.

Example

If the TwinCAT ADS WCF service should be reachable over the url http://www.beckhoff.com/TwinCAT/Ads/Wcf/TcAdsService you have to use the following command to create a certificate with TcAdsWcfCertGen:

```
TcAdsWWcfCertGen.exe www.beckhoff.com
```

This will generate a certificate with the common name www.beckhoff.com which can be used for SSL encoding.

5 TwinCAT.Ads.Wcf

5.1 TcAdsService

5.1.1 Interfaces

5.1.1.1 ITcAdsServiceSimplex

The ITcAdsServiceSimplex interface provides access to the features which are available with the simplex endpoints of the TwinCAT Ads WCF Service.

Methods and Properties

ITcAdsServiceSimplex Methods	Description
Read1 [▸ 14]	Reads data from an ADS device and writes it to the given byte array reference.
Read2 [▸ 15]	Reads data from an ADS device and writes it to the given byte array reference.
Read3 [▸ 16]	Reads data from an ADS device and writes it to the given byte array reference.
ReadDeviceInfo [▸ 17]	Reads the identification and version number of an ADS server.
ReadState [▸ 17]	Reads the ADS status and the device status from an ADS server.
ReadWrite1 [▸ 18]	Writes data to an ADS device and then reads data from this device.
ReadWrite2 [▸ 18]	Writes data to an ADS device and then reads data from this device.
ReadWrite3 [▸ 19]	Writes data to an ADS device and then reads data from this device.
ReadWrite4 [▸ 20]	Writes data to an ADS device and then reads data from this device.
TryReadState [▸ 21]	Reads the ADS status and the device status from an ADS server. Unlike the ReadState method this method does not call an exception on failure. Instead an AdsErrorCode is returned. If the return value is equal to AdsErrorCode.NoError the call was successfull.
Write1 [▸ 21]	Writes data to an ADS device.
Write2 [▸ 22]	Writes data to an ADS device.
Write3 [▸ 23]	Writes data to an ADS device.
Write4 [▸ 23]	Writes data to an ADS device.
Write5 [▸ 24]	Writes data to an ADS device.
Write6 [▸ 25]	Writes data to an ADS device.
Write7 [▸ 25]	Writes data to an ADS device.
Write8 [▸ 26]	Writes data to an ADS device.
WriteControl1 [▸ 26]	Changes the ADS status and the device status of an ADS server.
WriteControl2 [▸ 27]	Writes the passed object value to the specified ADS symbol.The parameter type must have the same layout as the ADS symbol.

5.1.1.1.1 ITcAdsServiceSimplex.Read1

Reads data from an ADS device and writes it to the given byte array reference.

```
byte[ ] Read1 (
    string netId,
    int port,
    string variableName,
    int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
variableName	Name of the ADS symbol.
length	The length of the returned byte array.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.2 ITcAdsServiceSimplex.Read2

Reads data from an ADS device and writes it to the given byte array reference.

```
byte[ ] Read2 (
    string netId,
    int port,
    int indexGroup,
    int indexOffset,
    int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
length	The length of the returned byte array.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.3 ITcAdsServiceSimplex.Read3

Reads data from an ADS device and writes it to the given byte array reference.

```
byte[ ] Read3 (
    string netId,
    int port,
    long indexGroup,
    long indexOffset,
    int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
length	The length of the returned byte array.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.4 ITcAdsServiceSimplex.ReadDeviceInfo

Reads the identification and version number of an ADS server.

```
DeviceInfo ReadDeviceInfo (
    string netId,
    int port
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

Return Value

DeviceInfo struct containing the name of the device and the version information.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.5 ITcAdsServiceSimplex.ReadState

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

```
StateInfo ReadState (
    string netId,
    int port
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

Return Value

The ADS statue and device status.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.

Type	Description
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▸ 14]

5.1.1.1.6 ITcAdsServiceSimplex.ReadWrite1

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

```
byte[ ] ReadWrite1 (
    string netId,
    int port,
    int indexGroup,
    int indexOffset,
    int rdLength,
    byte [ ] wrbuffer
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
rdLength	The length of the returned byte array.
wrbuffer	Byte array that contains the data that should be written.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▸ 14]

5.1.1.1.7 ITcAdsServiceSimplex.ReadWrite2

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

```
byte[ ] ReadWrite2 (
    string netId,
    int port,
    long indexGroup,
    long indexOffset,
    int rdLength
    byte [ ] wrbuffer
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
rdLength	The length of the returned byte array.
wrbuffer	Byte array that contains the data that should be written.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.8 ITcAdsServiceSimplex.ReadWrite3

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

```
byte[ ] ReadWrite3 (
    string netId,
    int port,
    int indexGroup,
    int indexOffset,
    int rdLength,
    byte [ ] wrbuffer,
    int wrOffset,
    int wrLength
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
rdLength	The length of the returned byte array.
wrbuffer	Byte array that contains the data that should be written.
wrOffset	Offset of the data in wrbuffer.
wrLength	Length of the data in wrbuffer.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.9 ITcAdsServiceSimplex.ReadWrite4

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

```
byte[ ] ReadWrite4 (
    string netId,
    int port,
    long indexGroup,
    long indexOffset,
    int rdLength,
    byte [ ] wrbuffer,
    int wrOffset,
    int wrLength
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
rdLength	The length of the returned byte array.
wrbuffer	Byte array that contains the data that should be written.
wrOffset	Offset of the data in wrbuffer.
wrLength	Length of the data in wrbuffer.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.

Type	Description
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.10 ITcAdsServiceSimplex.TryReadState

Reads the ADS status and the device status from an ADS server. Unlike the ReadState method this method does not call an exception on failure. Instead an AdsErrorCode is returned. If the return value is equal to AdsErrorCode.NoError the call was successfull.

```
int TryReadState (
    string netId,
    int port,
    out StateInfo stateInfo
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
stateInfo	Output reference of type StateInfo which will receive the data.

Return Value

AdsErrorCode

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.11 ITcAdsServiceSimplex.Write1

Writes data to an ADS device.

```
void Write1 (
    string netId,
    int port ,
    string variableName,
    byte [ ] buffer
);
```

Parameters

netId	NetID of the target AMS Router.

port	Port of the target AMS Router.
variableName	Name of the ADS symbol.
buffer	Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▸ 14]

5.1.1.1.12 ITcAdsServiceSimplex.Write2

Writes data to an ADS device.

```
void Write2 (
    string netId,
    int port,
    int indexGroup,
    int indexOffset,
    byte [ ] buffer
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
buffer	Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▸ 14]

5.1.1.1.13 ITcAdsServiceSimplex.Write3

Writes data to an ADS device.

```
void Write3 (
    string netId,
    int port,
    long indexGroup,
    long indexOffset,
    byte [ ] buffer
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
buffer	Byte array that contains the data.

Requirements

Table 1: SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▸ 14]

5.1.1.1.14 ITcAdsServiceSimplex.Write4

Writes data to an ADS device.

```
void Write4 (
    string netId,
    int port,
    string variableName,
    byte [ ] buffer,
    int offset,
    int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
variableName	Name of the ADS symbol.
buffer	Byte array that contains the data.
offset	Offset of the data in buffer.
length	Length of the data in buffer.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.15 ITcAdsServiceSimplex.Write5

Writes data to an ADS device.

```
void Write5 (
     string netId,
     int port,
     int indexGroup,
     int indexOffset,
     byte [ ] buffer,
     int offset,
     int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
buffer	Byte array that contains the data.
offset	Offset of the data in buffer.
length	Length of the data in buffer.

Requirements

Table 2: SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.16 ITcAdsServiceSimplex.Write6

Writes data to an ADS device.

```
void Write6 (
     string netId,
     int port,
     long indexGroup,
     long indexOffset,
     byte [ ] buffer,
     int offset,
     int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
buffer	Byte array that contains the data.
offset	Offset of the data in buffer.
length	Length of the data in buffer.

SOAP Error Faults

Requirements

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.17 ITcAdsServiceSimplex.Write7

Writes data to an ADS device.

```
void Write7(
    string netId,
    int port,
    int indexGroup,
    int indexOffset);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.

indexOffset	Contains the index offset number of the requested ADS service.

SOAP Error Faults

Requirements

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.18 ITcAdsServiceSimplex.Write8

Writes data to an ADS device.

```
void Write8(
    string netId,
    int port,
    long indexGroup,
    long indexOffset);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.

SOAP Error Faults

Requirements

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▶ 14]

5.1.1.1.19 ITcAdsServiceSimplex.WriteControl1

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

```
 void WriteControl1 (
    string netId,
    int port,
    StateInfo stateInfo
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
stateInfo	New ADS status and device status.

SOAP Error Faults

Requirements

Type	Description
	AdsErrorFault [▸ 65]he AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▸ 14]

5.1.1.1.20 ITcAdsServiceSimplex.WriteControl2

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

```
 void WriteControl2 (
    string netId,
    int port,
    StateInfo stateInfo,
    byte [ ] buffer,
    int offset,
    int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
stateInfo	New ADS status and device status.
buffer	Byte array that contains the data that should be sent to the ADS device.
offset	Offset of the data in buffer.
length	Length of the data in buffer.

Requirements

Table 3: SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.

Type	Description
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceSimplex Interface [▸ 14]

5.1.1.2 ITcAdsServiceDuplex

The ITcAdsServiceDuplex interface provides access to the features which are available with the duplex endpoints of the TwinCAT Ads WCF Service.

Methods and Properties

ITcAdsServiceDuplex Methods	Description
SubscribeAdsNotificationEvent [▸ 30]	Subscribes to the AdsNotificationEventHandler of the associated TcAdsClient. Callbacks will be passed through the ITcAdsServiceCallbaks.AdsNotification [▸ 53] method.
SubscribeAdsNotificationErrorEvent [▸ 30]	Subscribes to the AdsNotificationErrorEventHandler of the associated TcAdsClient. Callbacks will be passed through the ITcAdsServiceCallbaks.AdsNotificationError [▸ 54] method.
SubscribeAdsStateChangedEvent [▸ 30]	Subscribes to the AdsStateChangedEventHandler of the associated TcAdsClient. Callbacks will be passed through the ITcAdsServiceCallbaks.AdsStateChanged [▸ 54] method.
SubscribeAdsSymbolVersionChangedEvent [▸ 31]	Subscribes to the AdsSymbolVersionChangedEventHandler of the associated TcAdsClient. Callbacks will be passed through the ITcAdsServiceCallbaks.AdsSymbolVersionChanged [▸ 54] method.
SubscribeAmsRouterNotificationEvent [▸ 31]	Subscribes to the AmsRouterNotificationEventHandler of the associated TcAdsClient. Callbacks will be passed through the ITcAdsServiceCallbaks.AmsRouterNotification [▸ 55] method.
AddDeviceNotification1 [▸ 32]	Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▸ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▸ 30] method was called successfully.
AddDeviceNotification2 [▸ 33]	Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▸ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▸ 30] method was called successfully.
AddDeviceNotification3 [▸ 34]	Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▸ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▸ 30] method was called successfully.
AddBufferedDeviceNotification1 [▸ 35]	Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▸ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▸ 30] method was called successfully.

ITcAdsServiceDuplex Methods	Description
AddBufferedDeviceNotification2 [▶ 35]	Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▶ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▶ 30] method was called successfully.
AddBufferedDeviceNotification3 [▶ 36]	Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▶ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▶ 30] method was called successfully.
DeleteDeviceNotification [▶ 37]	Deletes an existing notification.
GetTimeout [▶ 38]	Gets the timeout for the associated TcAdsClient.
SetTimeout [▶ 38]	Sets the timeout for the associated TcAdsClient.
Read1 [▶ 39]	Reads data from an ADS device and writes it to the given byte array reference.
Read2 [▶ 40]	Reads data from an ADS device and writes it to the given byte array reference.
Read3 [▶ 40]	Reads data from an ADS device and writes it to the given byte array reference.
ReadDeviceInfo [▶ 41]	Reads the identification and version number of an ADS server.
ReadState [▶ 41]	Reads the ADS status and the device status from an ADS server.
ReadWrite1 [▶ 42]	Writes data to an ADS device and then reads data from this device.
ReadWrite2 [▶ 43]	Writes data to an ADS device and then reads data from this device.
ReadWrite3 [▶ 44]	Writes data to an ADS device and then reads data from this device.
ReadWrite4 [▶ 44]	Writes data to an ADS device and then reads data from this device.
TryReadState [▶ 45]	Reads the ADS status and the device status from an ADS server. Unlike the ReadState method this method does not call an exception on failure. Instead an AdsErrorCode is returned. If the return value is equal to AdsErrorCode.NoError the call was successfull.
Write1 [▶ 46]	Writes data to an ADS device.
Write2 [▶ 46]	Writes data to an ADS device.
Write3 [▶ 47]	Writes data to an ADS device.
Write4 [▶ 48]	Writes data to an ADS device.
Write5 [▶ 48]	Writes data to an ADS device.
Write6 [▶ 49]	Writes data to an ADS device.
Write7 [▶ 50]	Writes data to an ADS device.
Write8 [▶ 50]	Writes data to an ADS device.
WriteControl1 [▶ 51]	Changes the ADS status and the device status of an ADS server.
WriteControl2 [▶ 52]	Writes the passed object value to the specified ADS symbol.The parameter type must have the same layout as the ADS symbol.

5.1.1.2.1 ITcAdsServiceDuplex.SubscribeAdsNotificationEvent

Subscribes to the AdsNotificationEventHandler of the associated TcAdsClient. Callbacks will be passed through the ITcAdsServiceCallbaks.AdsNotification [▸ 53] method.

```
void SubscribeAdsNotificationEvent (
    string netId,
    int port
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

SOAP Error Faults

Type	Description
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28] \| ITcAdsServiceDuplex Interface [▸ 28] \| ITcAdsServiceCallbacks Interface [▸ 52]

5.1.1.2.2 ITcAdsServiceDuplex.SubscribeAdsNotificationErrorEvent

Subscribes to the AdsNotificationErrorEventHandler of the associated TcAdsClient. Callbacks will be passed through the ITcAdsServiceCallbaks.AdsNotificationError [▸ 54] method.

```
void SubscribeAdsNotificationErrorEvent (
    string netId,
    int port
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

SOAP Error Faults

Type	Description
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28] \| ITcAdsServiceDuplex Interface [▸ 28] \| ITcAdsServiceCallbacks Interface [▸ 52]

5.1.1.2.3 ITcAdsServiceDuplex.SubscribeAdsStateChangedEvent

Subscribes to the AdsStateChangedEventHandler of the associated TcAdsClient. Callbacks will be passed through the ITcAdsServiceCallbaks.AdsStateChanged [▸ 54] method.

```
void SubscribeAdsStateChangedEvent (
    string netId,
    int port
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

SOAP Error Faults

Type	Description
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceCallbacks Interface [▶ 52]

5.1.1.2.4 ITcAdsServiceDuplex.SubscribeAdsSymbolVersionChangedEvent

Subscribes to the AdsSymbolVersionChangedEventHandler of the associated TcAdsClient. Callbacks will be passed through the ITcAdsServiceCallbaks.AdsSymbolVersionChanged [▶ 54] method.

```
void SubscribeAdsSymbolVersionChangedEvent (
    string netId,
    int port
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

SOAP Error Faults

Type	Description
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceCallbacks Interface [▶ 52]

5.1.1.2.5 ITcAdsServiceDuplex.SubscribeAmsRouterNotificationEvent

Subscribes to the AmsRouterNotificationEventHandler of the associated TcAdsClient. Callbacks will be passed through the ITcAdsServiceCallbaks.AmsRouterNotification [▶ 55] method.

```
void SubscribeAmsRouterNotificationEvent (
    string netId,
    int port
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

SOAP Error Faults

Type	Description
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28] \| ITcAdsServiceDuplex Interface [▸ 28] \| ITcAdsServiceCallbacks Interface [▸ 52]

5.1.1.2.6 ITcAdsServiceDuplex.AddDeviceNotification1

Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▸ 53] method if the ITcAdsServiceDuplex .SubsicbeAdsNotificationEvent [▸ 30] method was called successfully.

```
int AddDeviceNotification1 (
    string netId,
    int port,
    string variableName,
    int length,
    AdsTransMode transMode,
    int cycleTime,
    int maxDelay,
    object userData );
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
variableName	The name of the ADS symbol.
length	The length of the byte array buffer which will be returned by the AdsNotification callback.
transMode	Specifies if the event should be fired cyclically or only if the variable has changed.
cycleTime	The ADS server checks whether the variable has changed after this time interval. Unit is ms.
maxDelay	The AdsNotification event is fired at the latest when this time has elapsed. The unit is ms.
userData	This object can be used to store user specific data. Only data types which can be serialized are allowed.

Return Value

The handle of the notification.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.

Type	Description
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28] | ITcAdsServiceDuplex Interface [▸ 28] | ITcAdsServiceCallbacks Interface [▸ 52]

5.1.1.2.7 ITcAdsServiceDuplex.AddDeviceNotification2

Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▸ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▸ 30] method was called successfully.

```
int AddDeviceNotification2 (
    string netId,
    int port,
    int indexGroup,
    int indexOffset,
    int length,
    AdsTransMode transMode,
    int cycleTime,
    int maxDelay,
    object userData );
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
length	The length of the byte array buffer which will be returned by the AdsNotification callback.
transMode	Specifies if the event should be fired cyclically or only if the variable has changed.
cycleTime	The ADS server checks whether the variable has changed after this time interval. Unit is ms.
maxDelay	The AdsNotification event is fired at the latest when this time has elapsed. The unit is ms.
userData	This object can be used to store user specific data. Only data types which can be serialized are allowed.

Return Value

The handle of the notification.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceCallbacks Interface [▶ 52]

5.1.1.2.8 ITcAdsServiceDuplex.AddDeviceNotification3

Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▶ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▶ 30] method was called successfully.

```
int AddDeviceNotification3 (
    string netId,
    int port,
    long indexGroup,
    long indexOffset,
    int length,
    AdsTransMode transMode,
    int cycleTime,
    int maxDelay,
    object userData
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
length	The length of the byte array buffer which will be returned by the AdsNotification callback.
transMode	Specifies if the event should be fired cyclically or only if the variable has changed.
cycleTime	The ADS server checks whether the variable has changed after this time interval. Unit is ms.
maxDelay	The AdsNotification event is fired at the latest when this time has elapsed. The unit is ms.
userData	This object can be used to store user specific data. Only data types which can be serialized are allowed.

Return Value

The handle of the notification.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceCallbacks Interface [▶ 52]

5.1.1.2.9 ITcAdsServiceDuplex.AddBufferedDeviceNotification1

Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▶ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▶ 30] method was called successfully.

```
int AddBufferedDeviceNotification1(
    string netId,
    int port,
    string variableName,
    int length,
    AdsTransMode transMode,
    int cycleTime,
    int maxDelay,
    int maxValues,
    object userData
);
```

Parameters

maxValues	This values defines when a buffered notification is fired.
userData	This object can be used to store user specific data. Only data types which can be serialized are allowed.

Return Value

The handle of the notification.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceCallbacks Interface [▶ 52]

5.1.1.2.10 ITcAdsServiceDuplex.AddBufferedDeviceNotification2

Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▶ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▶ 30] method was called successfully.

```
int AddBufferedDeviceNotification2 (
    string netId,
    int port,
    int indexGroup,
    int indexOffset,
    int length,
    AdsTransMode transMode,
    int cycleTime,
    int maxDelay,
    int maxValues,
    object userData
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
length	The length of the byte array buffer which will be returned by the AdsNotification callback.
transMode	Specifies if the event should be fired cyclically or only if the variable has changed.
cycleTime	The ADS server checks whether the variable has changed after this time interval. Unit is ms.
maxDelay	The AdsNotification event is fired at the latest when this time has elapsed. The unit is ms.
maxValues	This values defines when a buffered notification is fired.
userData	This object can be used to store user specific data. Only data types which can be serialized are allowed.

Return Value

The handle of the notification.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceCallbacks Interface [▶ 52]

5.1.1.2.11 ITcAdsServiceDuplex.AddBufferedDeviceNotification3

Connects a variable to the associated TcAdsClient. The notification will be passed through the ITcAdsServiceCallbaks.AdsNotification [▶ 53] method if the ITcAdsServiceDuplex .SubscibeAdsNotificationEvent [▶ 30] method was called successfully.

```
int AddDeviceNotification3 (
    string netId,
    int port,
    long indexGroup,
    long indexOffset,
    int length,
    AdsTransMode transMode,
    int cycleTime,
    int maxDelay,
    int maxValues,
    object userData
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
length	The length of the byte array buffer which will be returned by the AdsNotification callback.
transMode	Specifies if the event should be fired cyclically or only if the variable has changed.
cycleTime	The ADS server checks whether the variable has changed after this time interval. Unit is ms.
maxDelay	The AdsNotification event is fired at the latest when this time has elapsed. The unit is ms.
maxValues	This values defines when a buffered notification is fired.
userData	This object can be used to store user specific data. Only data types which can be serialized are allowed.

Return Value

The handle of the notification.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28] | ITcAdsServiceDuplex Interface [▸ 28] | ITcAdsServiceCallbacks Interface [▸ 52]

5.1.1.2.12 ITcAdsServiceDuplex.DeleteDeviceNotification

Deletes an existing notification.

```
void DeleteDeviceNotification (
    string netId,
    int port,
    int notificationHandle );
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
notificationHandle	Handle of the notification.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceCallbacks Interface [▶ 52]

5.1.1.2.13 ITcAdsServiceDuplex.GetTimeout

Gets the timeout for the associated TcAdsClient.

```
int GetTimeout (
    string netId,
    int port
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

Return Value

The actual timeout value in milliseconds.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsNotificationClientNotFoundErrorFault [▶ 66]	The AdsNotificationClientNotFoundErrorFault is returned if a TcAdsClient object for AdsNotification through passing is not found in the NotificationClient cache of a wcf duplex context.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceDuplex Interface [▶ 28] | ITcAdsServiceCallbacks Interface [▶ 52]

5.1.1.2.14 ITcAdsServiceDuplex.SetTimeout

Sets the timeout for the associated TcAdsClient.

```
void SetTimeout (
    string netId,
    int port,
    int value
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
value	The timeout value in milliseconds.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsNotificationClientNotFoundErrorFault [▸ 66]	The AdsNotificationClientNotFoundErrorFault is returned if a TcAdsClient object for AdsNotification through passing is not found in the NotificationClient cache of a wcf duplex context.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28] | ITcAdsServiceDuplex Interface [▸ 28] | ITcAdsServiceCallbacks Interface [▸ 52]

5.1.1.2.15 ITcAdsServiceDuplex.Read1

Reads data from an ADS device and writes it to the given byte array reference.

```
byte[ ] Read1 (
    string netId,
    int port,
    string variableName,
    int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
variableName	Name of the ADS symbol.
length	The length of the returned byte array.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.16 ITcAdsServiceDuplex.Read2

Reads data from an ADS device and writes it to the given byte array reference.

```
byte[ ] Read2 (
    string netId,
    int port,
    int indexGroup,
    int indexOffset,
    int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
length	The length of the returned byte array.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.17 ITcAdsServiceDuplex.Read3

Reads data from an ADS device and writes it to the given byte array reference.

```
byte[ ] Read3 (
    string netId,
    int port,
    long indexGroup,
    long indexOffset,
    int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
length	The length of the returned byte array.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.18 ITcAdsServiceDuplex.ReadDeviceInfo

Reads the identification and version number of an ADS server.

```
DeviceInfo ReadDeviceInfo (
    string netId,
    int port
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

Return Value

DeviceInfo struct containing the name of the device and the version information.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.19 ITcAdsServiceDuplex.ReadState

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

```
StateInfo ReadState (
    string netId,
    int port
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.

Return Value

The ADS statue and device status.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28]

5.1.1.2.20 ITcAdsServiceDuplex.ReadWrite1

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

```
byte[ ] ReadWrite1 (
    string netId,
    int port,
    int indexGroup,
    int indexOffset,
    int rdLength,
    byte [ ] wrbuffer
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
rdLength	The length of the returned byte array.
wrbuffer	Byte array that contains the data that should be written.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28]

5.1.1.2.21 ITcAdsServiceDuplex.ReadWrite2

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

```
byte[ ] ReadWrite2 (
    string netId,
    int port,
    long indexGroup,
    long indexOffset,
    int rdLength,
    byte [ ] wrbuffer
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
rdLength	The length of the returned byte array.
wrbuffer	Byte array that contains the data that should be written.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28]

5.1.1.2.22 ITcAdsServiceDuplex.ReadWrite3

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

```
byte[ ] ReadWrite3 (
    string netId,
    int port,
    int indexGroup,
    int indexOffset,
    int rdLength,
    byte [ ] wrbuffer,
    int wrOffset,
    int wrLength
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
rdLength	The length of the returned byte array.
wrbuffer	Byte array that contains the data that should be written.
wrOffset	Offset of the data in wrbuffer.
wrLength	Length of the data in wrbuffer.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28]

5.1.1.2.23 ITcAdsServiceDuplex.ReadWrite4

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

```
byte [ ] ReadWrite4 (
    string netId,
    int port,
    long indexGroup,
    long indexOffset,
    int rdLength,
    byte [ ] wrbuffer,
    int wrOffset,
    int wrLength
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
rdLength	The length of the returned byte array.
wrbuffer	Byte array that contains the data that should be written.
wrOffset	Offset of the data in wrbuffer.
wrLength	Length of the data in wrbuffer.

Return Value

Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.24 ITcAdsServiceDuplex.TryReadState

Reads the ADS status and the device status from an ADS server. Unlike the ReadState method this method does not call an exception on failure. Instead an AdsErrorCode is returned. If the return value is equal to AdsErrorCode.NoError the call was successfull.

```
int TryReadState (
    string netId,
    int port,
    out StateInfo stateInfo
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
stateInfo	Output reference of type StateInfo which will receive the data.

Return Value

AdsErrorCode

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.

Type	Description
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28]

5.1.1.2.25 ITcAdsServiceDuplex.Write1

Writes data to an ADS device.

```
void Write1 (
    string netId,
    int port ,
    string variableName,
    byte [ ] buffer
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
variableName	Name of the ADS symbol.
buffer	Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28]

5.1.1.2.26 ITcAdsServiceDuplex.Write2

Writes data to an ADS device.

```
void Write2 (
    string netId,
    int port,
    int indexGroup,
    int indexOffset,
    byte [ ] buffer
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
buffer	Byte array that contains the data.

SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.27 ITcAdsServiceDuplex.Write3

Writes data to an ADS device.

```
void Write3 (
    string netId,
    int port,
    long indexGroup,
    long indexOffset,
    byte [ ] buffer
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
buffer	Byte array that contains the data.

Requirements

Table 4: SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28]

5.1.1.2.28 ITcAdsServiceDuplex.Write4

Writes data to an ADS device.

```
void Write4 (
    string netId,
    int port,
    string variableName,
    byte [ ] buffer,
    int offset,
    int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
variableName	Name of the ADS symbol.
buffer	Byte array that contains the data.
offset	Offset of the data in buffer.
length	Length of the data in buffer.

SOAP Error Faults

Type	Description
AdsErrorFault [▶ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▶ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▶ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▶ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▶ 28]

5.1.1.2.29 ITcAdsServiceDuplex.Write5

Writes data to an ADS device.

```
void Write5 (
     string netId,
     int port,
     int indexGroup,
     int indexOffset,
     byte [ ] buffer,
     int offset,
     int length
);
```

Parameters

netId	NetID of the target AMS Router.

port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
buffer	Byte array that contains the data.
offset	Offset of the data in buffer.
length	Length of the data in buffer.

Requirements

Table 5: SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.30 ITcAdsServiceDuplex.Write6

Writes data to an ADS device.

```
void Write6 (
     string netId,
     int port,
     long indexGroup,
     long indexOffset,
     byte [ ] buffer,
     int offset,
     int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.
buffer	Byte array that contains the data.
offset	Offset of the data in buffer.
length	Length of the data in buffer.

SOAP Error Faults

Requirements

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.

Type	Description
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
ArgumentErrorFault [▸ 66]	The ArgumentErrorFault is returned if there is a failure in an argument of a method.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.31 ITcAdsServiceDuplex.Write7

Writes data to an ADS device.

```
void Write7(
    string netId,
    int port,
    int indexGroup,
    int indexOffset);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.

SOAP Error Faults

Requirements

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.32 ITcAdsServiceDuplex.Write8

Writes data to an ADS device.

```
void Write8(
    string netId,
    int port,
    long indexGroup,
    long indexOffset);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
indexGroup	Contains the index group number of the requested ADS service.
indexOffset	Contains the index offset number of the requested ADS service.

SOAP Error Faults

Requirements

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.33 ITcAdsServiceDuplex.WriteControl1

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

```
 void WriteControl1 (
    string netId,
    int port,
    StateInfo stateInfo
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
stateInfo	New ADS status and device status.

SOAP Error Faults

Requirements

Type	Description
	AdsErrorFault [▸ 65]he AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.2.34 ITcAdsServiceDuplex.WriteControl2

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

```
 void WriteControl2 (
    string netId,
    int port,
    StateInfo stateInfo,
    byte [ ] buffer,
    int offset,
    int length
);
```

Parameters

netId	NetID of the target AMS Router.
port	Port of the target AMS Router.
stateInfo	New ADS status and device status.
buffer	Byte array that contains the data that should be sent to the ADS device.
offset	Offset of the data in buffer.
length	Length of the data in buffer.

Requirements

Table 6: SOAP Error Faults

Type	Description
AdsErrorFault [▸ 65]	The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
AdsErrorErrorFault [▸ 65]	The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.
UnexpectedErrorFault [▸ 66]	The UnexpectedErrorFault is returned if an unexpected Error occurs.

See Also

ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.3 ITcAdsServiceCallbacks

The ITcAdsServiceCallbacks interface provides access to the callback methods of the duplex endpoints.

Methods and Properties

ITcAdsServiceCallbacks Methods	Description
AdsNotification [▸ 53]	Passes through AdsNotificationEventHandler callback of the associated TcAdsClient. To subscribe to the AdsNotificationEventHandler the ITcAdsServiceDuplex.SubscribeAdsNotificationEvent [▸ 30] method has to be called.
AdsNotificationError [▸ 54]	Passes through AdsNotificationErrorEventHandler callback of the associated TcAdsClient. To subscribe to the AdsNotificationErrorEventHandler the ITcAdsServiceDuplex.SubscribeAdsNotificationErrorEvent [▸ 54] method has to be called.

ITcAdsServiceCallbacks Methods	Description
AdsStateChanged [▶ 54]	Passes through AdsStateChangedEventHandler callback of the associated TcAdsClient. To subscribe to the AdsStateChangedEventHandler the ITcAdsServiceDuplex.SubscribeAdsStateChangedEvent [▶ 30] method has to be called.
AdsSymbolVersionChanged [▶ 54]	Passes through AdsSymbolVersionChangedEventHandler callback of the associated TcAdsClient. To subscribe to the AdsSymbolVersionChangedEventHandler the ITcAdsServiceDuplex.SubscribeAdsSymbolVersionChangedEvent [▶ 54] method has to be called.
AmsRouterNotification [▶ 55]	Passes through AmsRouterNotificationEventHandler callback of the associated TcAdsClient. To subscribe to the AmsRouterNotificationEventHandler the ITcAdsServiceDuplex.SubscribeAmsRouterNotificationEvent [▶ 55] method has to be called.
HeartBeat [▶ 55]	Used by the TwinCAT ADS WCF Service to test if the client application is still reachable. If the HeartBeat callback fails, the client will be set to a Faulted State.

5.1.1.3.1 ITcAdsServiceCallbacks.AdsNotification

Passes through AdsNotificationEventHandler callback of the associated TcAdsClient.
To subscribe to the AdsNotificationEventHandler the ITcAdsServiceDuplex.SubscribeAdsNotificationEvent [▶ 30] method has to be called.

```
void AdsNotification (
    string netId,
    int port,
    byte [ ] buffer,
    int length,
    int notificationHandle,
    int offset,
    long timeStamp,
    object userData
);
```

Parameters

netId	NetID of the AMS Router.
port	Port of the AMS Router.
buffer	Byte array that holds the notification data. *Notice* **If you use buffered notifications, you will receive only one callback which contains all data.**
length	The length of the data in buffer.
notificationHandle	The handle of the connection.
offset	The offset of the data in buffer.
timeStamp	The timestamp when the notification was raised.
userData	This object is passed by to AddDeviceNotification methods and can be used to store data.

See Also

ITcAdsServiceCallbacks Interface [▶ 52] | ITcAdsServiceDuplex Interface [▶ 28]

5.1.1.3.2 ITcAdsServiceCallbacks.AdsNotificationError

Passes through AdsNotificationErrorEventHandler callback of the associated TcAdsClient.
To subscribe to the AdsNotificationErrorEventHandler the ITcAdsServiceDuplex.SubscribeAdsNotificationErrorEvent [▸ 54] method has to be called.

```
 void AdsNotificationError (
    string netId,
    int port,
    Exception error
);
```

Parameters

netId	NetID of the AMS Router.
port	Port of the AMS Router.
error	Exception that was caught while handling notifications.

See Also

ITcAdsServiceCallbacks Interface [▸ 52] | ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.3.3 ITcAdsServiceCallbacks.AdsStateChanged

Passes through AdsStateChangedEventHandler callback of the associated TcAdsClient.
To subscribe to the AdsStateChangedEventHandler the ITcAdsServiceDuplex.SubscribeAdsStateChangedEvent [▸ 30] method has to be called.

```
void AdsStateChanged (
    string netId,
    int port,
    StateInfo state
);
```

Parameters

netId	NetID of the AMS Router.
port	Port of the AMS Router.
state	Current state of the ADS device.

See Also

ITcAdsServiceCallbacks Interface [▸ 52] | ITcAdsServiceDuplex Interface [▸ 28]

5.1.1.3.4 ITcAdsServiceCallbacks.AdsSymbolVersionChanged

Passes through AdsSymbolVersionChangedEventHandler callback of the associated TcAdsClient.
To subscribe to the AdsSymbolVersionChangedEventHandler the ITcAdsServiceDuplex.SubscribeAdsSymbolVersionChangedEventHandler [▸ 54] method has to be called.

```
void AdsSymbolVersionChanged (
    string netId,
    int port
);
```

Parameters

netId	NetID of the AMS Router.
port	Port of the AMS Router.

See Also

ITcAdsServiceCallbacks Interface [▶ 52] | TcAdsServiceDuplex Interface [▶ 28]

5.1.1.3.5 ITcAdsServiceCallbacks.AmsRouterNotification

Passes through AmsRouterNotificationEventHandler callback of the associated TcAdsClient.
To subscribe to the AmsRouterNotificationEventHandler the ITcAdsServiceDuplex.SubscribeAmsRouterNotificationEventHandler [▶ 55] method has to be called.

```
void AmsRouterNotification (
    string netId,
    int port,
    AmsRouterState state
);
```

Parameters

netId	NetID of the AMS Router.
port	Port of the AMS Router.
state	Current state of the Ams Router.

See Also

ITcAdsServiceCallbacks Interface [▶ 52] | ITcAdsServiceDuplex Interface [▶ 28]

5.1.1.3.6 ITcAdsServiceCallbacks.HeartBeat

Used by the TwinCAT ADS WCF Service to test if the client application is still reachable.
The HeartBeat will be send every 5 seconds.
If the HeartBeat callback fails, the client will be set to a Faulted State.
It can also be used to detect if the server is still alive.

```
void HeartBeat( );
```

See Also

ITcAdsServiceCallbacks Interface [▶ 52] | ITcAdsServiceDuplex Interface [▶ 28]

5.1.2 Enumerations

5.1.2.1 AdsErrorCode

```
public enum AdsErrorCode
{
    NoError = 0,
    InternalError = 1,
    NoRTime = 2,
    LockedMemoryError = 3,
    MailBoxError = 4,
    WrongHMsg = 5,
    TargetPortNotFound = 6,
    TargetMachineNotFound = 7,
    UnknownCommandID = 8,
    BadTaskID = 9,
    NoIO = 10,
    UnknwonAmsCommand = 11,
    Win32Error = 12,
    PortNotConnected = 13,
    InvalidAmsLength = 14,
    InvalidAmsNetID = 15,
    LowInstallLevel = 16,
    NoDebug = 17,
    PortDisabled = 18,
    PortConnected = 19,
    AmsSyncWin32Error = 20,
    SyncTimeOut = 21,
    AmsSyncAmsError = 22,
    AmsSyncNoIndexMap = 23,
    InvalidAmsPort = 24,
    NoMemory = 25,
    TCPSendError = 26,
    HostUnreachable = 27,
    NoLockedMemory = 1280,
    MailboxFull = 1282,
    DeviceError = 1792,
    DeviceServiceNotSupported = 1793,
    DeviceInvalidGroup = 1794,
    DeviceInvalidOffset = 1795,
    DeviceInvalidAccess = 1796,
    DeviceInvalidSize = 1797,
    DeviceInvalidData = 1798,
    DeviceNotReady = 1799,
    DeviceBusy = 1800,
    DeviceInvalidContext = 1801,
    DeviceNoMemory = 1802,
    DeviceInavlidParam = 1803,
    DeviceNotFound = 1804,
    DeviceSyntaxError = 1805,
    DeviceIncompatible = 1806,
    DeviceSymbolNotFound = 1808,
    DeviceSymbolVersionInvalid = 1809,
    DeviceInvalidState = 1810,
    DeviceTransModeNotSupported = 1811,
    DeviceNotifyHandleInvalid = 1812,
    DeviceClientUnknown = 1813,
    DeviceNoMoreHandles = 1814,
    DeviceInvalidWatchsize = 1815,
    DeviceNotInitialized = 1816,
    DeviceTimeOut = 1817,
    DeviceNoInterface = 1818,
    DeviceInvalidInterface = 1819,
    DeviceInvalidCLSID = 1820,
    DeviceInvalidObjectID = 1821,
    ClientError = 1856,
    ClientInvalidParameter = 1857,
    ClientListEmpty = 1858,
    ClientVaraiableInUse = 1859,
    ClientDuplicateInvokeID = 1860,
    ClientSyncTimeOut = 1861,
    ClientW32OR = 1862,
    ClientTimeoutInvalid = 1863,
    ClientPortNotOpen = 1864,
    ClientNoAmsAddr = 1865,
    ClientSyncInternal = 1872,
    ClientAddHash = 1873,
    ClientRemoveHash = 1874,
    ClientNoMoreSymbols = 1875,
    ClientSyncResInvalid = 1876,
    ClientSyncPortLocked = 1877,
    ClientQueueFull = 32768,
}
```

Members

Member Name	Description
NoError	No Error. Error code: 0(0x000).
InternalError	Internal Error. Error code: 1(0x001).
NoRTime	No Rtime. Error code: 2(0x002).
LockedMemoryError	Allocation locked memory error.

Member Name	Description
	Error code: 3(0x003).
MailBoxError	Insert mailbox error. Error code: 4(0x004).
WrongHMsg	Wrong receive HMSG. Error code: 5(0x005).
TargetPortNotFound	Target port not found. Error code: 6(0x006).
TargetMachineNotFound	Target machine not found. Error code: 7(0x007).
UnknownCommandID	Unknown command ID. Error code: 8(0x008).
BadTaskID	Bad task ID. Error code: 9(0x009).
NoIO	No IO. Error code: 10(0x00A).
UnknownAmsCommand	Unknown AMS command. Error code: 11(0x00B).
Win32Error	Win 32 error. Error code: 12(0x00C).
PortNotConnected	Port is not connected. Error code: 13(0x00D).
InvalidAmsLength	Invalid AMS length. Error code: 14(0x00E).
InvalidAmsNetID	Invalid AMS Net ID. Error code: 15(0x00F).
LowInstallLevel	Low Installation level. Error code: 16(0x010).
NoDebug	No debug available. Error code: 17(0x011).
PortDisabled	Port disabled. Error code: 18(0x012).
PortConnected	Port is already connected. Error code: 19(0x013).
AmsSyncWin32Error	AMS Sync Win32 error. Error code: 20(0x014).
SyncTimeOut	AMS Sync timeout. Error code: 21(0x015).
AmsSyncAmsError	AMS Sync AMS error Error code: 22(0x016).
AmsSyncNoIndexMap	AMS Sync no index map. Error code: 23(0x017).
InvalidAmsPort	Invalid AMS port. Error code: 24(0x018).
NoMemory	No memory. Error code: 25(0x019).
TCPSendError	TCP send error. Error code: 26(0x01A).

Member Name	Description
HostUnreachable	Host unreachable. Error code: 27(0x1B).
NoLockedMemory	Router: no locked memory. Error code: 1280(0x500).
MailboxFull	Router: mailbox full. Error code: 1282(0x501).
DeviceError	error class <device error"> Error code: 1792(0x700).
DeviceServiceNotSupported	Service is not supported by server. Error code: 1793(0x701).
DeviceInvalidGroup	Invalid index group. Error code: 1794(0x702).
DeviceInvalidOffset	Invalid index offset. Error code: 1795(0x703).
DeviceInvalidAccess	Reading/writing not permitted. Error code: 1796(0x704).
DeviceInvalidSize	Parameter size not correct. Error code: 1797(0x705).
DeviceInvalidData	Invalid parameter value(s). Error code: 1798(0x706).
DeviceNotReady	Device is not in a ready state. Error code: 1799(0x707).
DeviceBusy	Device is busy. Error code: 1800(0x708).
DeviceInvalidContext	Invalid context (must be in Windows). Error code: 1801(0x709).
DeviceNoMemory	Out of memory. Error code: 1802(0x70a).
DeviceInavlidParam	Invalid parameter value(s). Error code: 1803(0x70b).
DeviceNotFound	Not found (files, ...). Error code: 1804(0x70c).
DeviceSyntaxError	Syntax error in command or file. Error code: 1805(0x70d).
DeviceIncompatible	Objects do not match. Error code: 1806(0x70e).
DeviceExists	Object already exists. Error code: 1807(0x70f).
DeviceSymbolNotFound	Symbol not found. Error code: 1808(0x7010).
DeviceSymbolVersionInvalid	Symbol version is invalid. Error code: 1809(0x711).
DeviceInvalidState	Server is not i a valid state. Error code: 1810(0x712).
DeviceTransModeNotSupported	ADS transmode is not supported. Error code: 1811(0x713).
DeviceNotifyHandleInvalid	Notification handle is invalid.

Member Name	Description
	Error code: 1812(0x714).
DeviceClientUnknown	Notification client not registered. Error code: 1813(0x715).
DeviceNoMoreHandles	No more notification handles. Error code: 1814(0x716).
DeviceInvalidWatchsize	Size for watch to big. Error code: 1815(0x717).
DeviceNotInitialized	Device is not initialized. Errocr code: 1818(0x718).
DeviceTimeOut	Device has a timeout. Error code: 1817(0x719).
DeviceNoInterface	Query interface has failed. Error code: 1818(0x71A).
DeviceInvalidInterface	Wrong interface required. Error code: 1819(0x71B).
DeviceInvalidCLSID	Class ID is invalid. Error code: 1820(0x71C).
DeviceInvalidObjectID	Object ID is invalid. Error code: 1821(0x71D).
ClientError	Error class <client error> Error code: 1856(0x740).
ClientInvalidParameter	Parameter at service is invalid. Error code: 1857(0x741).
ClientListEmpty	Polling list is empty. Error code: 1858(0x742).
ClientVaraiableInUse	Variable connection is already in use. Error code: 1859(0x743).
ClientDuplicateInvokeID	Invoke ID already in use. Error code: 1860(0x744).
ClientSyncTimeOut	Timeout has elapsed. Error code: 1861(x745).
ClientW32OR	Error in win32 subsystem. Error code: 1862(0x746).
ClientTimeoutInvalid	Timeout value is invalid. Error code: 1863(0x747).
ClientPortNotOpen	Ads port is not opened. Error code: 1864(0x748).
ClientNoAmsAddr	No AMS addr. Error code: 1865(0x749).
ClientSyncInternal	An internal in ADS sync has occured. Error code: 1872(0x750).
ClientAddHash	Hash table overflow. Error code: 1873(0x751).
ClientRemoveHash	There are no more symbols in the hash table. Error code: 1874(0x752).
ClientNoMoreSymbols	There are no more symbols in cache. Error code: 1875(0x753).

Member Name	Description
ClientSyncResInvalid	An invalid response has been received. Error code: 1876(0x754).
ClientSyncPortLocked	Sync port is locked. Error code: 1877(0x755).
ClientQueueFull	Client queue is full

5.1.2.2 AdsReservedIndexGroups

```
public enum AdsReservedIndexGroups
{
    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,
}
```

Members

Member Name	Description
PlcRWMB	
PlcRWMX	
PlcRWRB	
PlcRWDB	
SymbolTable	
SymbolName	
SymbolValue	
SymbolHandleByName	
SymbolValueByName	
SymbolValueByHandle	
SymbolReleaseHandle	
SymbolInfoByName	
SymbolVersion	
SymbolInfoByNameEx	
SymbolDownload	
SymbolUpload	
SymbolUploadInfo	
SymbolNote	
IOImageRWIB	

Member Name	Description
IOImageRWIX	
IOImageRWOB	
IOImageRWOX	
IOImageClearI	
IOImageClearO	
DeviceData	

5.1.2.3 AdsReservedIndexOffset

```
public enum AdsReservedIndexOffset
{
    DeviceDataAdsState = 0,
    DeviceDataDeviceState = 2,
    DeviceDataConfigID = 4,
}
```

Members

Member Name	Description
DeviceDataAdsState	
DeviceDataDeviceState	
DeviceDataConfigID	

5.1.2.4 AdsState

```
public enum AdsState
{
    Invalid = 0,
    Idle = 1,
    Reset = 2,
    Init = 3,
    Start = 4,
    Run = 5,
    Stop = 6,
    SaveConfig = 7,
    LoadConfig = 8,
    PowerFailure = 9,
    PowerGood = 10,
    Error = 11,
    Shutdown = 12,
    Suspend = 13,
    Resume = 14,
    Config = 15,
    Reconfig = 16,
    Maxstates = 17,
}
```

Members

Member Name	Description
Invalid	invalided state
Idle	idle state
Reset	reset state
Init	initialized
Start	started
Run	running
Stop	stopped
SaveConfig	saved configuration

Member Name	Description
LoadConfig	load configuration
PowerFailure	power failure
PowerGood	power good
Error	error state
Shutdown	shutting down
Suspend	suspended
Resume	resumed
Config	system is in config mode
Reconfig	system should restart in config mode
Maxstates	

5.1.2.5 AdsTransMode

```
public enum AdsTransMode
{
    Cyclic = 3,
    OnChange = 4,
}
```

Members

Member Name	Description
Cyclic	The AdsSyncNotification-Event is fired cyclically.
OnChange	The AdsSyncNotification-Event is fired when the data changes.

5.1.2.6 AmsPort

```
public enum AmsPort
{
    Router = 1,
    Debugger = 2,
    Logger = 100,
    EventLog = 110,
    R0_Realtime = 200,
    R0_Trace = 290,
    R0_IO = 300,
    R0_NC = 500,
    R0_NCSAF = 501,
    R0_NCSVB = 511,
    R0_ISG = 550,
    R0_CNC = 600,
    R0_LINE = 700,
    R0_PLC = 800,
    PlcRuntime1 = 801,
    PlcRuntime2 = 811,
    PlcRuntime3 = 821,
    PlcRuntime4 = 831,
    CamshaftController = 900,
    R0_CAMTOOL = 950,
    R0_USER = 2000,
    R3_CTRLPROG = 10000,
    SystemService = 10000,
    R3_SYSCTRL = 10001,
    R3_SYSSAMPLER = 10100,
    R3_TCPRAWCONN = 10200,
    R3_TCPIPSERVER = 10201,
    R3_SYSMANAGER = 10300,
    R3_SMSSERVER = 10400,
    R3_MODBUSSERVER = 10500,
    R3_S7SERVER = 10600,
    R3_PLCCONTROL = 10800,
    R3_NCCTRL = 11000,
    R3_NCINTERPRETER = 11500,
    R3_STRECKECTRL = 12000,
    R3_CAMCTRL = 13000,
    R3_SCOPE = 14000,
    R3_SINECH1 = 15000,
    R3_CONTROLNET = 16000,
    R3_OPCSERVER = 17000,
    R3_OPCCLIENT = 17500,
    USEDEFAULT = 65535,
}
```

Members

Member Name	Description
Router	AmsRouter
Debugger	AmsDebugger
Logger	Logger
EventLog	Event Logger
R0_Realtime	R0 Realtime
R0_Trace	R0 Trace
R0_IO	R0 IO
R0_NC	NC (R0)
R0_NCSAF	R0 Satzausführung
R0_NCSVB	R0 Satzvorbereitung
R0_ISG	R0 ISG
R0_CNC	R0 CNC
R0_LINE	R0 Line
R0_PLC	R0 PLC
PlcRuntime1	PLC RuntimeSystem 1
PlcRuntime2	PLC RuntimeSystem 2
PlcRuntime3	PLC RuntimeSystem 3
PlcRuntime4	PLC RuntimeSystem 4
CamshaftController	Camshaft Controller (R0)
R0_CAMTOOL	R0 CAM Tool
R0_USER	R0 User
SystemService	System Service (AMSPORT_R3_SYSSERV, 10000)
R3_CTRLPROG	
R3_SYSCTRL	
R3_SYSSAMPLER	
R3_TCPRAWCONN	
R3_TCPIPSERVER	
R3_SYSMANAGER	
R3_SMSSERVER	
R3_MODBUSSERVER	
R3_S7SERVER	
R3_PLCCONTROL	
R3_NCCTRL	
R3_NCINTERPRETER	
R3_STRECKECTRL	
R3_CAMCTRL	
R3_SCOPE	
R3_SINECH1	
R3_CONTROLNET	

Member Name	Description
R3_OPCSERVER	
R3_OPCCLIENT	
USEDEFAULT	

5.1.2.7 AmsRouterState

```
public enum AmsRouterState
{
    Stop = 0,
    Start = 1,
    Removed = 2,
}
```

Members

Member Name	Description
Stop	Ams Router is stopped.
Start	Ams Router is started.
Removed	Ams Router has been removed.

5.1.3 Structures

5.1.3.1 DeviceInfo

The DeviceInfo structure contains the name and the version information of the device.

```
public struct DeviceInfo
{
    public string Name { get; set; }
    public AdsVersion Version { get; set; }
}
```

StateInfo Properties	Description
Name	Gets or sets the name of the device.
Version	Gets or sets the version information.

5.1.3.2 StateInfo

The StateInfo structure contains the Ads state and device state.

```
public struct StateInfo
{
    public AdsState AdsState { get; set; }
    public short DeviceState { get; set; }
}
```

StateInfo Proper-ties	Description
AdsState	Gets or sets the Ads state.

StateInfo Properties	**Description**
DeviceState	Gets or sets the device state.

5.1.3.3 AdsVersion

The DeviceInfo structure contains the name and the version information of the device.

```
public struct AdsVersion
{
    public int Build { get; set; }
    public byte Revision { get; set; }
    public byte Version { get; set; }
}
```

StateInfo Properties	**Description**
Build	Gets or sets the build number.
Revision	Gets or sets the revision number.
Version	Gets or sets the version number.

5.1.4 SOAP Error Faults

5.1.4.1 AdsErrorFault

The AdsErrorFault is returned if an AdsException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.

Table 7: Properties

Property	**Description**
string Message	Description of the Error which causes the ErrorFault.

5.1.4.2 AdsErrorErrorFault

The AdsErrorErrorFault is returned if an AdsErrorException is thrown by a TcAdsClient object in a method of the TwinCAT ADS WCF service.

Table 8: Properties

Property	**Description**
string Message	Description of the Error which causes the ErrorFault.
AdsErrorCode [▸ 55] ErrorCode	The AdsErrorCode of the AdsErrorException.

5.1.4.3 ArgumentErrorFault

The ArgumentErrorFault is returned if there is a failure in an argument of a method.

Table 9: Properties

Property	Description
string Message	Description of the Error which causes the ErrorFault.

5.1.4.4 AdsNotificationClientNotFoundErrorFault

The AdsNotificationClientNotFoundErrorFault is returned if a TcAdsClient object for AdsNotification through passing is not found in the NotificationClient cache of a wcf duplex context.

Table 10: Properties

Property	Description
string Message	Description of the Error which causes the ErrorFault.

5.1.4.5 UnexpectedErrorFault

The UnexpectedErrorFault is returned if an unexpected Error occurs.

Table 11: Properties

Property	Description
string Message	Description of the Error which causes the ErrorFault.

5.1.5 Endpoints

Name	Description
UnsecNetTcp [▸ 68]	duplex / unsecured
SecNetTcp [▸ 68]	duplex / secured
UnsecWsDualHttp [▸ 67]	duplex / unsecured
SecWsDualHttp [▸ 67]	duplex / secured
UnsecPollingDuplex3 [▸ 68]	duplex / unsecured
SecPollingDuplex3 [▸ 68]	duplex / secured
UnsecPollingDuplex4 [▸ 69]	duplex / unsecured
SecPollingDuplex4 [▸ 68]	duplex / secured
UnsecBasicHttp [▸ 67]	simplex / unsecured

Name	Description
SecBasicHttp [▶ 67]	simplex / secured
UnsecWebHttp [▶ 67]	simplex / unsecured
SecWebHttp [▶ 67]	simplex / secured

See also

Safety references [▶ 13]

5.1.5.1 SecWebHttp

Address: https://[target_name]:8002/TwinCAT/Ads/Wcf/TcAdsService/SecWebHttp

The SecWebHttp endpoint provides secured access to the TcAdsService WCF service in a simplex context via http.

See also

Safety references [▶ 13]

5.1.5.2 UnsecWebHttp

Address: https://[target_name]:8002/TwinCAT/Ads/Wcf/TcAdsService/UnsecWebHttp

The UnsecWebHttp endpoint provides unsecured access to the TcAdsService WCF service in a simplex context via http.

See also

Safety references [▶ 13]

5.1.5.3 SecBasicHttp

Address: https://[target_name]:8002/TwinCAT/Ads/Wcf/TcAdsService/SecBasicHttp

The SecBasicHttp endpoint provides secured access to the TcAdsService WCF service in a simplex context via http.

See also

Safety references [▶ 13]

5.1.5.4 UnsecBasicHttp

Address: http://[target_name]:8003/TwinCAT/Ads/Wcf/TcAdsService/UnsecBasicHttp

The UnsecBasicHttp endpoint provides unsecured access to the TcAdsService WCF service in a simplex context via http.

See also

Safety references [▶ 13]

5.1.5.5 SecWsDualHttp

Address: https://[target_name]:8002/TwinCAT/Ads/Wcf/TcAdsService/SecWsDualHttp

The SecWsDualHttp endpoint provides secured access to the TcAdsService WCF service in a duplex context via http.

See also

Safety references [▶ 13]

5.1.5.6 UnsecWsDualHttp

Address: http://[target_name]:8003/TwinCAT/Ads/Wcf/TcAdsService/UnsecWsDualHttp

The UnsecWsDualHttp endpoint provides unsecured access to the TcAdsService WCF service in a duplex context via http.

See also

Safety references [▶ 13]

5.1.5.7 SecNetTcp

Address: net.tcp://[target_name]:4508/TwinCAT/Ads/Wcf/TcAdsService/SecNetTcp

The SecNetTcp endpoint provides secured access to the TcAdsService WCF service in a duplex context via tcp.

See also

Safety references [▶ 13]

5.1.5.8 UnsecNetTcp

Address: net.tcp://[target_name]:4508/TwinCAT/Ads/Wcf/TcAdsService/UnsecNetTcp

The UnsecNetTcp endpoint provides unsecured access to the TcAdsService WCF service in a duplex context via tcp.

See also

Safety references [▶ 13]

5.1.5.9 SecPollingDuplex3

Address: https://[target_name]:8002/TwinCAT/Ads/Wcf/TcAdsService/SecPollingDuplex3

The SecPollingDuplex3 endpoint provides secured access to the TcAdsService WCF service in a duplex context via http for Silverlight 3 applications.

See also

Safety references [▶ 13]

5.1.5.10 UnsecPollingDuplex3

Address: http://[target_name]:8003/TwinCAT/Ads/Wcf/TcAdsService/UnsecPollingDuplex3

The UnsecPollingDuplex3 endpoint provides unsecured access to the TcAdsService WCF service in a duplex context via http for Silverlight 3 applications.

See also

Safety references [▶ 13]

5.1.5.11 SecPollingDuplex4

Address: https://[target_name]:8002/TwinCAT/Ads/Wcf/TcAdsService/SecPollingDuplex4

The SecPollingDuplex4 endpoint provides secured access to the TcAdsService WCF service in a duplex context via http for Silverlight 4 applications.

See also

Safety references [▸ 13]

5.1.5.12 UnsecPollingDuplex4

Address: http://[target_name]:8003/TwinCAT/Ads/Wcf/TcAdsService/UnsecPollingDuplex4

The UnsecPollingDuplex endpoint provides unsecured access to the TcAdsService WCF service in a duplex context via http for Silverlight 4 applications.

See also

Safety references [▸ 13]

5.2 ClientAccessPolicyProvider

The ClientAccessPolicyProvider is a WCF (Windows Communication Foundation) service which provides access to the ClientAccessPolicy.xml via HTTP Get on the defined base addresses.

If a web client makes a call to a WCF service like the TcAdsService, it will ask for the ClientAccessPolicy.xml file to determine whether cross domain access to this resource is allowed or not.

Web clients connected over Net.Tcp are requesting the ClientAccessPolicy.xml on port 80.

If the Internet Information Services (IIS) are running, they will listening for ClientAccessPolicy.xml request on port 80.
In this case the ClientAccessPolicy.xml has to be placed in the wwwroot directory. The default path is: "*C:\Intepub\wwwroot\*".

If the Internet Information Services (IIS) are not installed the ClientAccessPolicyProvider has to be listening on port 80. Therefore the base address "http://localhost:80/" has to be added to the App.config of the TcAdsWcfHost.exe application. Thebase address of the ClientAccessPolicyProvider servicemustalso beadjusted if the default base addresses of the TcAdsService have changed.

5.2.1 ClientAccessPolicy.xml

The ClientAccessPolicy.xml is used by web clients to determine if cross domain access is allowed or not.

MSDN about ClientAccessPolicy.xml and Silverlight

The default ClientAccessPolicy.xml configuration of the TwinCAT ADS WCF.
The default configuration should not be used in production environments.

```
<?xml version="1.0" encoding="utf-8"?>
    <access-policy>
    <cross-domain-access>
        <policy>
        <allow-from http-request-headers="*">
            <domain uri="http://*"/>
            <domain uri="https://*"/>
        </allow-from>
        <grant-to>
            <resource path="/" include-subpaths="true"/>
            <socket-resource port="4502-4534" protocol="tcp"/>
        </grant-to>
        </policy>
    </cross-domain-access>
    </access-policy>
```

6 .NET Samples

Description
Create a WCF service reference in Visual Studio [▶ 11]
Create a WCF web reference in Visual Studio 2008 [▶ 77]

Microsoft .NET

Description
Accessing an array in the PLC [▶ 70]
Event driven reading [▶ 72]
Buffered Values [▶ 76]
Reading and writing string variables [▶ 76]
Handling SOAP ErrorFaults and specific WCF Exceptions [▶ 75]

Microsoft Compact Framework

Description
Accessing an array in the PLC [▶ 78]
Reading and writing string variables [▶ 80]
Handling SOAP ErrorFaults and specific WCF Exceptions [▶ 81]

Microsoft Silverlight

Description
Accessing an array in the PLC [▶ 82]
Event driven reading [▶ 84]
Reading and writing string variables [▶ 87]
Handling SOAP ErrorFaults and specific WCF Exceptions [▶ 88]

6.1 Accessing an array in the PLC

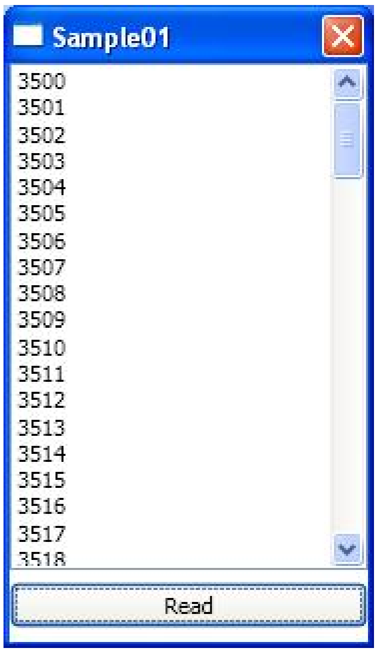
Requirements

Visual Studio 2010

TwinCAT ADS WCF 1.1.0.1

Description

Reading an array of integer variables from the PLC by the use of the TwinCAT ADS WCF service.
The TwinCAT.Ads DLL is used to read the data from the received byte array in an easy to handle way.



```
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
```

WPF (C#) application

```
using System.Windows;
using System.ServiceModel;
using TwinCAT.Ads;
using Sample01.TwinCAT.Ads.Wcf;

namespace Sample01
{
    public partial class MainWindow : Window
    {
    // If the netid string is empty, the TwinCAT ADS WCF service will use the TwinCAT Runtime on its
 host machine.
    private const string NETID = "";
    private const int PORT = 801;

    private TcAdsServiceSimplexClient wcfClient;

    public MainWindow ( )
    {
        InitializeComponent ( );

        // Initialize simplex client with BasicHttpBinding.
        wcfClient = new TcAdsServiceSimplexClient (
            new BasicHttpBinding( BasicHttpSecurityMode.None ),
            new EndpointAddress ( "http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService/
UnsecBasicHttp" )
        );

        // Open connection to the WCF service.
        wcfClient.Open ( );
    }

    private void btnRead_Click ( object sender, RoutedEventArgs e )
    {
        // Create buffer as byte array which recieves the data.
        int buffLen = 100 * 2;

        // Read array from plc.
         byte[] buffer =  wcfClient.Read1(NETID, PORT, "MAIN.PLCVar", buffLen);

        // Create AdsStream and AdsBinaryReader for buffer handling.
        AdsStream dataStream = new AdsStream ( buffer );
        AdsBinaryReader binRead = new AdsBinaryReader ( dataStream );

        lbArray.Items.Clear ( );
        dataStream.Position = 0;
        for ( int i = 0; i < 100; i++ )
        {
        lbArray.Items.Add ( binRead.ReadInt16( ) );
        }
    }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481886987/.zip

6.2 Event driven reading

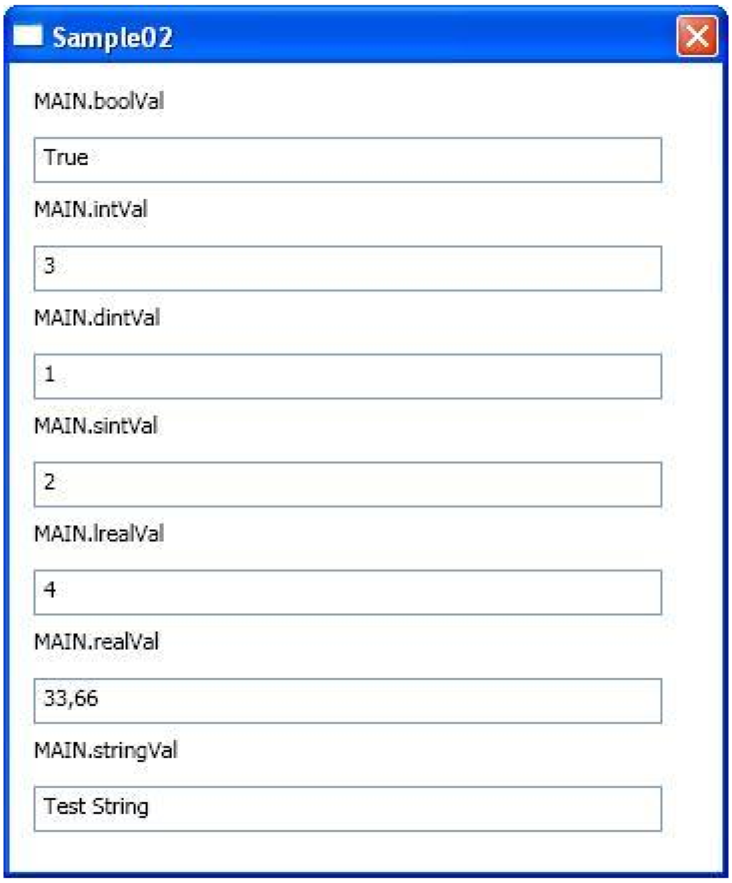
Requirements

Visual Studio 2010

TwinCAT ADS WCF 1.1.0.1

Description

Reading symbol data from the PLC by ADS notification callback on symbol change.
The TwinCAT.Ads DLL is used to read the data from the received byte array in an easy to handle way.



MAIN.boolVal

True

MAIN.intVal

3

MAIN.dintVal

1

MAIN.sintVal

2

MAIN.lrealVal

4

MAIN.realVal

33,66

MAIN.stringVal

Test String

WPF (C#) application

```
using System;
using System.Windows;
using TwinCAT.Ads;
using Sample02.TcAdsWcf;
using System.ServiceModel;
using System.Windows.Threading;
using System.Threading;

namespace Sample02
{
    // ConcurrencyMode.Multiple and UseSynchronizationContext = false prevent the GUI from Deadlocks
,
    // while waiting for a WCF response on the callback channel.
    [CallbackBehavior (
    ConcurrencyMode = ConcurrencyMode.Multiple,
    UseSynchronizationContext = false )]
    public partial class MainWindow : Window, ITcAdsServiceDuplexCallback
    {
     // If the netid string is empty, the TwinCAT ADS WCF service will use the TwinCAT Runtime on it
s host machine.
    private const string NETID = "";
    private const int PORT = 801;

    private TcAdsServiceDuplexClient wcfClient;
    private int[] handles = new int [ 7 ];

    public MainWindow ( )
    {
        InitializeComponent ( );

        // Initialize duplex client with NetTcpBinding.
        wcfClient = new TcAdsServiceDuplexClient (
        new InstanceContext ( this ),
        new NetTcpBinding ( SecurityMode.None ),
        new EndpointAddress ( "net.tcp://localhost:4508/TwinCAT/Ads/Wcf/TcAdsService/
UnsecNetTcp" ) );

        // Open connection to the WCF service.
        wcfClient.Open ( );

        // Subscribe to AdsNotification and AdsNotificationError events.
        wcfClient.SubscribeAdsNotificationEvent ( NETID, PORT );
        wcfClient.SubscribeAdsNotificationErrorEvent ( NETID, PORT );

        // Register device notifications.
        handles [ 0 ] = wcfClient.AddDeviceNotification1 (
        NETID, PORT, "MAIN.boolVal", 1, TcAdsWcf.AdsTransMode.OnChange, 100, 0, null );
        handles [ 1 ] = wcfClient.AddDeviceNotification1 (
        NETID, PORT, "MAIN.intVal", 2, TcAdsWcf.AdsTransMode.OnChange, 100, 0, null );
        handles [ 2 ] = wcfClient.AddDeviceNotification1 (
        NETID, PORT, "MAIN.dintVal", 4, TcAdsWcf.AdsTransMode.OnChange, 100, 0, null );
        handles [ 3 ] = wcfClient.AddDeviceNotification1 (
        NETID, PORT, "MAIN.sintVal", 1, TcAdsWcf.AdsTransMode.OnChange, 100, 0, null );
        handles [ 4 ] = wcfClient.AddDeviceNotification1 (
        NETID, PORT, "MAIN.lrealVal", 8, TcAdsWcf.AdsTransMode.OnChange, 100, 0, null );
        handles [ 5 ] = wcfClient.AddDeviceNotification1 (
        NETID, PORT, "MAIN.realVal", 4, TcAdsWcf.AdsTransMode.OnChange, 100, 0, null );
        handles [ 6 ] = wcfClient.AddDeviceNotification1 (
        NETID, PORT, "MAIN.stringVal", 50, TcAdsWcf.AdsTransMode.OnChange, 100, 0, null );
    }

    #region ITcAdsServiceDuplexCallback Members

    public void AdsNotification ( string netId, int port, byte [ ] buffer, int length, int notificat
ionHandle, int offset, long timeStamp, object userData )
    {
        // Create AdsStream and AdsBinaryReader for buffer handling.
        AdsStream adsStream = new AdsStream ( buffer );
        AdsBinaryReader binReader = new AdsBinaryReader ( adsStream );
        adsStream.Position = offset;

        // Choose action by notificationHandle value.
        // UI changes have to be invoked,
        // because the callback methods are called from a background thread.
        if ( notificationHandle == handles [ 0 ] )
        {
        string value = binReader.ReadBoolean ( ).ToString ( );
        Dispatcher.BeginInvoke (
            DispatcherPriority.Input,
            new ThreadStart ( ( ) =>
            {
            tbBool.Text = value;
            } ) );
        }
        else if ( notificationHandle == handles [ 1 ] )
        {
        string value = binReader.ReadInt16 ( ).ToString ( );
        Dispatcher.BeginInvoke (
            DispatcherPriority.Input,
            new ThreadStart ( ( ) =>
            {
            tbInt.Text = value;
            } ) );
        }
        else if ( notificationHandle == handles [ 2 ] )
        {
        string value = binReader.ReadInt32 ( ).ToString ( );
        Dispatcher.BeginInvoke (
            DispatcherPriority.Input,
            new ThreadStart ( ( ) =>
            {
            tbDint.Text = value;
            } ) );
        }
        else if ( notificationHandle == handles [ 3 ] )
        {
        string value = binReader.ReadByte ( ).ToString ( );
        Dispatcher.BeginInvoke (
            DispatcherPriority.Input,
            new ThreadStart ( ( ) =>
            {
            tbSint.Text = value;
            } ) );
        }
        else if ( notificationHandle == handles [ 4 ] )
        {
        string value = binReader.ReadDouble ( ).ToString ( );
        Dispatcher.BeginInvoke (
            DispatcherPriority.Input,
            new ThreadStart ( ( ) =>
            {
            tbLreal.Text = value;
            } ) );
        }
        else if ( notificationHandle == handles [ 5 ] )
        {
        string value = binReader.ReadSingle ( ).ToString ( );
        Dispatcher.BeginInvoke (
            DispatcherPriority.Input,
            new ThreadStart ( ( ) =>
            {
            tbReal.Text = value;
            } ) );
        }
        else if ( notificationHandle == handles [ 6 ] )
        {
        string value = binReader.ReadPlcString ( length );
        Dispatcher.BeginInvoke (
            DispatcherPriority.Input,
            new ThreadStart ( ( ) =>
            {
            tbString.Text = value;
            } ) );
        }
    }

    public void AdsNotificationError ( string netId, int port, Exception error )
    {
        MessageBox.Show ( error.Message );
    }

    public void AdsStateChanged ( string netId, int port, TcAdsWcf.StateInfo state )
    {
        // Not needed
    }

    public void AdsSymbolVersionChanged ( string netId, int port )
    {
        // Not needed
    }

    public void AmsRouterNotification ( string netId, int port, TcAdsWcf.AmsRouterState state )
    {
        // Not needed
    }

    public void HeartBeat ( )
    {
        // Not needed
    }

    #endregion
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481888395/.zip

6.3 Handling SOAP ErrorFaults and specific WCF Exceptions

Requirements

Visual Studio 2010

TwinCAT ADS WCF 1.1.0.1

C# console application

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using TcAdsWcf_Sample09.TwinCAT.Ads.Wcf;
using System.ServiceModel;

namespace TcAdsWcf_Sample09
{
    class Program
    {
    static void Main ( string [ ] args )
    {
        // Initialize simplex client with BasicHttpBinding.
        TcAdsServiceSimplexClient wcfClient = new TcAdsServiceSimplexClient (
            new BasicHttpBinding ( BasicHttpSecurityMode.None ),
            new EndpointAddress ( "http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService/
UnsecBasicHttp" )
        );

        // Open connection to the WCF service.
        wcfClient.Open ( );

        try
        {
        byte[] buffer = wcfClient.Read1("", 801, "BAD_VARNAME", 2);
        }
        catch ( FaultException fe )
        {
        // Will be raised if an ErrorFault was raised by a method of TwinCAT ADS WCF.
        if ( fe.GetType ( ).Equals (
            typeof ( FaultException<ArgumentErrorFault> ) ) )
        {
            FaultException<ArgumentErrorFault> feArg =
            fe as FaultException<ArgumentErrorFault>;

            Console.WriteLine ( "ErrorFault: ArgumentErrorFault" );
            Console.WriteLine ( feArg.Message );
        }

        if ( fe.GetType ( ).Equals (
            typeof ( FaultException<AdsErrorFault> ) ) )
        {
            FaultException<AdsErrorFault> feAds =
            fe as FaultException<AdsErrorFault>;

            Console.WriteLine ( "ErrorFault: AdsErrorFault" );
            Console.WriteLine ( feAds.Message );
        }

        if ( fe.GetType ( ).Equals (
            typeof ( FaultException<AdsErrorErrorFault> ) ) )
        {
            FaultException<AdsErrorErrorFault> feAdsErr =
            fe as FaultException<AdsErrorErrorFault>;

            Console.WriteLine ( "ErrorFault: AdsErrorErrorFault" );
            Console.WriteLine ( feAdsErr.Message );
            Console.WriteLine ( "ErrorCode: " + feAdsErr.Detail.ErrorCode );
        }

        if ( fe.GetType ( ).Equals (
            typeof ( FaultException<UnexpectedErrorFault> ) ) )
        {
            FaultException<UnexpectedErrorFault> feUnexpected =
            fe as FaultException<UnexpectedErrorFault>;

            Console.WriteLine ( "ErrorFault: UnexpectedErrorFault" );
            Console.WriteLine ( feUnexpected.Message );
        }
        }
        catch ( CommunicationException ce )
        {
        // Will be raised if there is a communication problem with TwinCAT ADS WCF.
        Console.WriteLine ( "CommunicationException: " );
        Console.WriteLine ( ce.Message );
        }
        catch ( TimeoutException te )
        {
        // Will be raised if a requested timed out.
        Console.WriteLine ( "TimeoutException: " );
        Console.WriteLine ( te.Message );
        }
        catch ( Exception ex )
        {
        Console.WriteLine ( "Exception: " );
        Console.WriteLine ( ex.Message );
        }

        // Close connection to TwinCAT ADS WCF.
        wcfClient.Close ( );
        Console.ReadKey();
    }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481889803/.zip

6.4 Reading and writing string variables

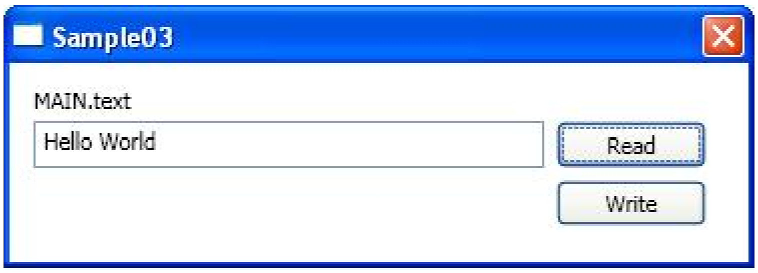
Requirements

Visual Studio 2010

TwinCAT ADS WCF 1.1.0.1

Description

Read a string from a plc variable and display it in the visualization or write a string from the visualization to a plc variable.
The TwinCAT.Ads DLL is used to read the data from the received byte array, or to write the data into the byte array which will be send to the service, in an easy-to-handle way.



WPF (C#) application

```
using System.ServiceModel;
using System.Windows;
using TcAdsWcf_Sample03.TwinCAT.Ads.Wcf;
using TwinCAT.Ads;

namespace TcAdsWcf_Sample03
{
    public partial class MainWindow : Window
    {
    // If the netid string is empty, the TwinCAT ADS WCF service will use the TwinCAT Runtime on its
```

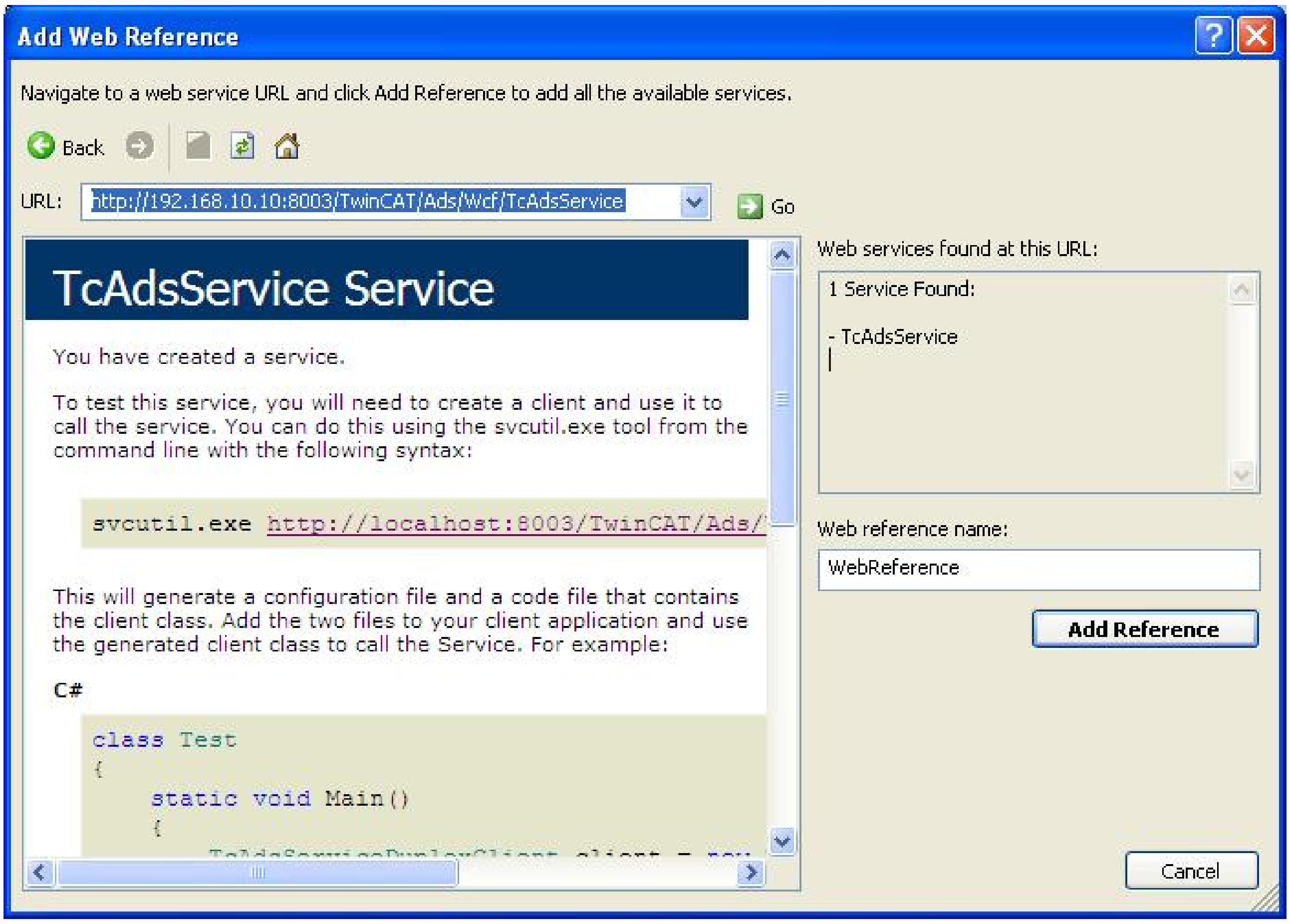
```
 host machine.
    private const string NETID = "";
    private const int PORT = 801;

    private TcAdsServiceSimplexClient wcfClient;

    public MainWindow ( )
    {
        InitializeComponent ( );

        // Initialize simplex client with BasicHttpBinding.
        wcfClient = new TcAdsServiceSimplexClient (
        new BasicHttpBinding ( BasicHttpSecurityMode.None ),
        new EndpointAddress ( "http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService/
UnsecBasicHttp" ) );

        // Open connection to the WCF service.
        wcfClient.Open ( );
    }

    private void BtnRead_Click ( object sender, RoutedEventArgs e )
    {
        // Read the data from the plc into a byte array buffer.
        byte[] buffer = wcfClient.Read1(NETID, PORT, "MAIN.text", 30);

        // Initialize an AdsStream and an AdsBinaryWriter to read the data from the buffer.
        AdsStream stream = new AdsStream ( buffer );
        AdsBinaryReader binReader = new AdsBinaryReader ( stream );

        string text = binReader.ReadPlcString ( 30 );
        TbText.Text = text;
    }

    private void BtnWrite_Click ( object sender, RoutedEventArgs e )
    {
        // Create a byte array buffer.
        // Length of text +1 because of zero termination.
        byte[] buffer = new byte[ TbText.Text.Length + 1 ];

        // Initialize an AdsStream and an AdsBinaryWriter to write the text to the buffer.
        AdsStream stream = new AdsStream ( buffer );
        AdsBinaryWriter binWriter = new AdsBinaryWriter ( stream );
        binWriter.WritePlcString ( TbText.Text, TbText.Text.Length + 1 );

        // Send the buffer.
        wcfClient.Write1 ( NETID, PORT, "MAIN.text", buffer );
    }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481891211/.zip

6.5 CE

6.5.1 Create a WCF web reference in Visual Studio 2008

In order to create a WCF web reference in Visual Studio 2008 you must choose the command *Add Web Reference...* under the *Project* menu.
This opens the Add Web Reference dialog:


Add Web Reference
Navigate to a web service URL and click Add Reference to add all the available services.
Back
URL:
Go
Web services found at this URL:
Web reference name:
Add Reference
Cancel

URL: http://192.168.10.10:8003/TwinCAT/Ads/Wcf/TcAdsService

TcAdsService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://localhost:8003/TwinCAT/Ads/
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

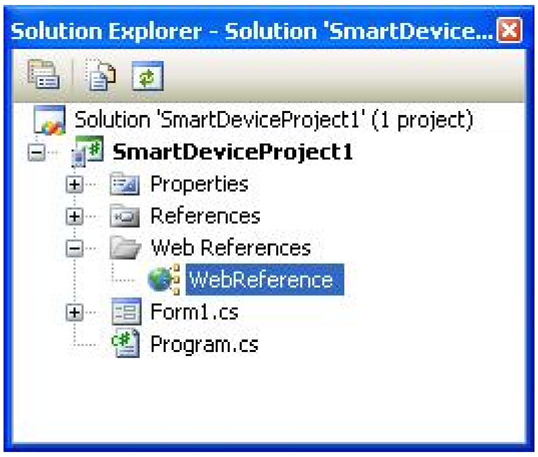
```
class Test
{
    static void Main()
    {
```

1 Service Found:

- TcAdsService

Web reference name: WebReference

In the dialog enter the http or the https based base address of the TcAdsService and press the *GO* button. This will load the service meta data and create the needed client code.
After that the web reference will be listed in the Solution Explorer with the chosen namespace.


Solution Explorer - Solution 'SmartDevice...
Solution 'SmartDeviceProject1' (1 project)
SmartDeviceProject1
Properties
References
Web References
WebReference
Form1.cs
Program.cs

6.5.2 Accessing an array in the PLC

Requirements

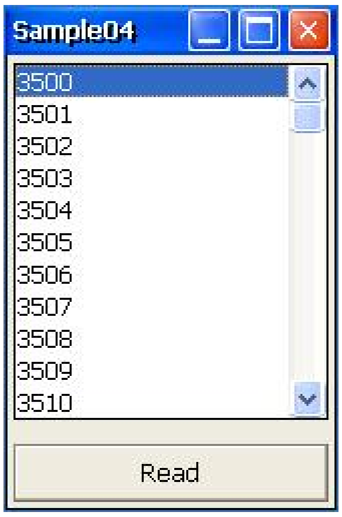
Visual Studio 2008

Target Windows CE Device

TwinCAT ADS WCF 1.1.0.1

Description

Reading an array of integer variables from the PLC using the TwinCAT ADS WCF service.
The TwinCAT.Ads DLL (Compat Framework) is used to read the data from the received byte array in an easy-to-handle way.



Compact Framework 3.5 (C#) application

```
using System;
using System.Windows.Forms;
using TcAdsWcf_Sample04.TwinCAT.Ads.Wcf;
using TwinCAT.Ads;

namespace TcAdsWcf_Sample04
{
    public partial class Form1 : Form
    {
    private const string NETID = "10.1.128.80.1.1";
    private const int PORT = 801;

    private UnsecBasicHttpEndpoint wcfClient;

    public Form1 ( )
    {
        InitializeComponent ( );

        // Initialize the WCF client.
        wcfClient = new UnsecBasicHttpEndpoint ( );
        wcfClient.Url = "http://10.1.128.80:8003/TwinCAT/Ads/Wcf/TcAdsService/UnsecBasicHttp";
    }

    private void btnRead_Click ( object sender, EventArgs e )
    {
        // Read the whole plc array into the byte array buffer.
        int buffLen = 100 * 2;
        byte[] buffer = wcfClient.Read1(NETID, PORT, true, "MAIN.PLCVar", buffLen, true );

        // Create an AdsStream and an AdsBinaryReader to read the data from the buffer.
        AdsStream stream = new AdsStream ( buffer );
        AdsBinaryReader binReader = new AdsBinaryReader ( stream );
        stream.Position = 0;

        lbArray.Items.Clear ( );
        for ( int i = 0; i < 100; i++ )
        {
        lbArray.Items.Add ( binReader.ReadInt16 ( ).ToString ( ) );
        }
    }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481892619/.zip

6.5.3 Reading and writing string variables

Requirements

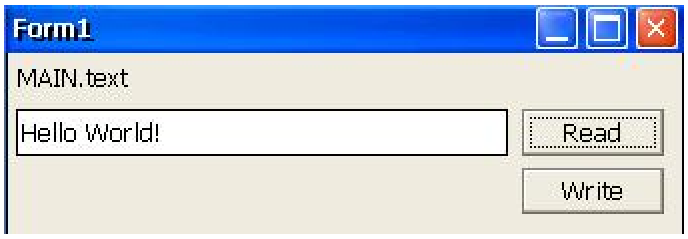
Visual Studio 2008

Target Windows CE Device

TwinCAT ADS WCF 1.1.0.1

Description

Read a string from a plc variable and display it in the visualization or write a string from the visualization to a plc variable.
The TwinCAT.Ads DLL (Compact Framework) is used to read the data from the received byte array, or to write the data into the byte array which will be send to the service, in an easy to handle way.



Compact Framework 3.5 (C#) application

```
using System;
using System.Windows.Forms;
using TcAdsWcf_Sample05.TwinCAT.Ads.Wcf;
using TwinCAT.Ads;

namespace TcAdsWcf_Sample05
{
    public partial class Form1 : Form
    {
    private const string NETID = "10.1.128.80.1.1";
    private const int PORT = 801;

    private UnsecBasicHttpEndpoint wcfClient;

    public Form1 ( )
    {
        InitializeComponent ( );

        // Initialize the WCF client.
        wcfClient = new UnsecBasicHttpEndpoint ( );
        wcfClient.Url = "http://10.1.128.80:8003/TwinCAT/Ads/Wcf/TcAdsService/UnsecBasicHttp";
    }

    private void btnRead_Click ( object sender, EventArgs e )
    {
        // Read the whole plc array into the byte array buffer.
        byte[] buffer = wcfClient.Read1 (NETID, PORT, true, "MAIN.text", 30, true);

        // Create an AdsStream and an AdsBinaryReader to read the data from the buffer.
        AdsStream stream = new AdsStream ( buffer );
        AdsBinaryReader binReader = new AdsBinaryReader ( stream );

        tbText.Text = binReader.ReadPlcString ( 30 );
    }

    private void btnWrite_Click ( object sender, EventArgs e )
    {
        // Create a byte array buffer.
        // Length of text +1 because of zero termination.
        byte [ ] buffer = new byte [ tbText.Text.Length + 1 ];

        // Initialize an AdsStream and an AdsBinaryWriter to write the text to the buffer.
        AdsStream stream = new AdsStream ( buffer );
        AdsBinaryWriter binWriter = new AdsBinaryWriter ( stream );
        binWriter.WritePlcString ( tbText.Text, tbText.Text.Length + 1 );
        // Send the buffer.
        wcfClient.Write1 ( NETID, PORT, true, "MAIN.text", buffer );
    }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481894027/.zip

6.5.4 Handling SOAP ErrorFaults and specific WCF Exceptions

Requirements

Visual Studio 2008

Target Windows CE Device

TwinCAT ADS WCF 1.1.0.1

Compact Framework 3.5 (C#) application

```
using System;
using System.Windows.Forms;
using TcAdsWcf_Sample11.TwinCAT.Ads.WCF;
using System.Web.Services.Protocols;
using System.Xml;

namespace TcAdsWcf_Sample11
{
    public partial class Form1 : Form
    {
    private UnsecBasicHttpEndpoint wcfClient;

    public Form1 ( )
    {
        InitializeComponent ( );

        // Initialize the WCF client.
        wcfClient = new UnsecBasicHttpEndpoint ( );
        wcfClient.Url = "http://10.1.128.80:8003/TwinCAT/Ads/Wcf/TcAdsService/UnsecBasicHttp";
    }

    private void button1_Click ( object sender, EventArgs e )
    {
        try
        {
        // Error provocation
        byte [ ] buffer = wcfClient.Read1 ( "", 801, true, "BAD_VARNAME", 2, true);
        }
        catch ( SoapException soapEx )
        {
        if ( soapEx.Code.Name.Equals ( "AdsErrorFault" ) )
        {
            MessageBox.Show ( "AdsErrorFault: \n" + soapEx.Message );
        }

        if ( soapEx.Code.Name.Equals ( "AdsErrorErrorFault" ) )
        {
            // Parse details of serialized AdsErrorErrorFault object.
            //  Create XmlNamespaceManager and add needed namespaces.
            XmlNamespaceManager nsMgr = new XmlNamespaceManager ( soapEx.Detail.OwnerDocument.NameTa
ble );
            nsMgr.AddNamespace ( "type", "http://beckhoff.com/TwinCAT/Ads/Wcf/Types" );

            // Get the value of the ErrorCode property.
            string errCodeVal = string.Empty;
            XmlNode errCodeNode = soapEx.Detail.SelectSingleNode (
            "./type:AdsErrorErrorFault/type:ErrorCode", nsMgr );

            errCodeVal = errCodeNode.InnerText;

            MessageBox.Show ( string.Format (
            "AdsErrorErrorFault: \n{0}\n\n ErrorCode: {1}",
            soapEx.Message, errCodeVal ) );
        }
        if ( soapEx.Code.Name.Equals ( "ArgumentErrorFault" ) )
        {
            MessageBox.Show ( "ArgumentErrorFault: \n" + soapEx.Message );
        }

        if ( soapEx.Code.Name.Equals ( "UnexpectedErrorFault" ) )
        {
            MessageBox.Show ( "UnexpectedErrorFault: \n" + soapEx.Message );
        }
        }
        catch ( Exception ex )
        {
        MessageBox.Show ( "Exception: \n" + ex.Message );
        }
    }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481895435/.zip

6.6 Silverlight

6.6.1 Accessing an array in the PLC

Requirements

Visual Studio 2010

Silverlight 4 Tools

TwinCAT ADS WCF 1.1.0.1

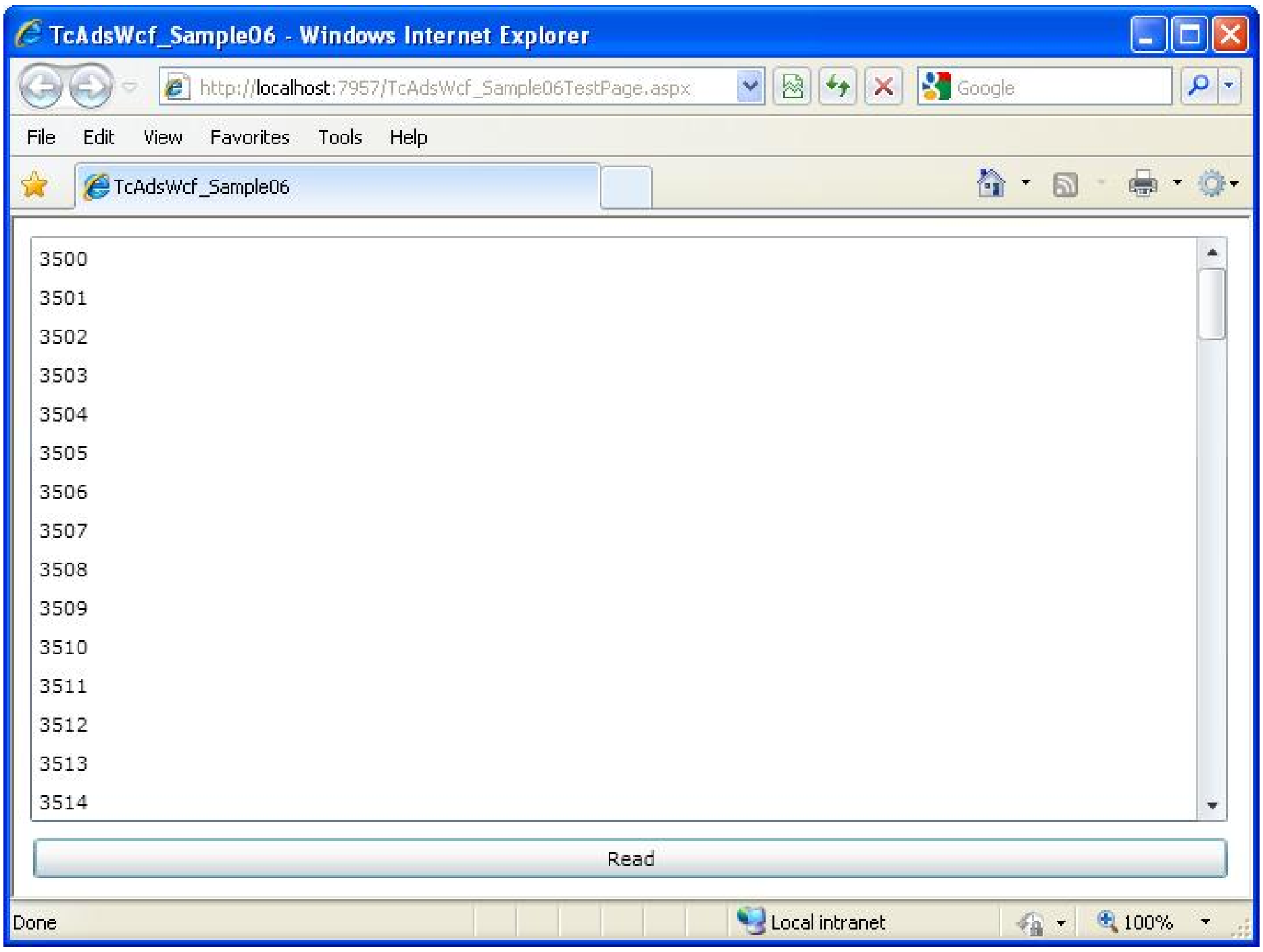
Description

Reading an array of integer variables from the PLC by the use of the TwinCAT ADS WCF service.
The TwinCAT.Ads.SL DLL (Silverlight) is used to read the data from the received byte array in an easy to handle way.

While using the WCF service within a Silverlight application, all methods of the used context interface are appended with the postfix Async.
And there is a extra callback method for each method with the postfix Completed too.

Example: Read1 -> Read1Async, Read1Completed

This is important because silverlight allows asynchronous communication only. A read action by the use of the Read1 method is requested by the use of the Read1Async method and the response will raise the Read1Completed callback.


TcAdsWcf_Sample06 - Windows Internet Explorer
http://localhost:7957/TcAdsWcf_Sample06TestPage.aspx
Google
File Edit View Favorites Tools Help
TcAdsWcf_Sample06
Read
Done
Local intranet
100%

```
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
```

Silverlight 4 application

```
using System;
using System.ServiceModel;
using System.Windows;
using System.Windows.Controls;
using TcAdsWcf_Sample06.TwinCAT.Ads.Wcf;
using TwinCAT.Ads.SL;

namespace TcAdsWcf_Sample06
{
    public partial class MainPage : UserControl
    {
    // If the netid string is empty, the TwinCAT ADS WCF service will use the TwinCAT Runtime on its
 host machine.
    private const string NETID = "";
    private const int PORT = 801;

    private TcAdsServiceSimplexClient wcfClient;

    public MainPage ( )
    {
        InitializeComponent ( );

        // Initialize WCF service client.
        wcfClient = new TcAdsServiceSimplexClient (
        new BasicHttpBinding ( ),
        new EndpointAddress ( "http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService/
UnsecBasicHttp" ) );

        // Register for Read1Completed Event
        wcfClient.Read1Completed += new EventHandler<Read1CompletedEventArgs1>(wcfClient_Read1Comple
ted);
    }

    private void wcfClient_Read1Completed(object sender, Read1CompletedEventArgs1 e)
    {
        // Create AdsStream and AdsBinaryReader for buffer handling.
        AdsStream stream = new AdsStream(e.Result);
        AdsBinaryReader binReader = new AdsBinaryReader(stream);
        stream.Position = 0;

        lbArray.Items.Clear();
        for (int i = 0; i < 100; i++)
        {
        lbArray.Items.Add(binReader.ReadInt16());
        }
    }

    private void btnRead_Click ( object sender, RoutedEventArgs e )
    {
        // Read array.
        int buffLen = 100 * 2;
        wcfClient.Read1Async(NETID, PORT, "MAIN.PLCVar", buffLen, null);
    }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481896843/.zip

6.6.2 Event driven reading

Requirements

Visual Studio 2010

Silverlight 4 Tools

TwinCAT ADS WCF 1.1.0.1

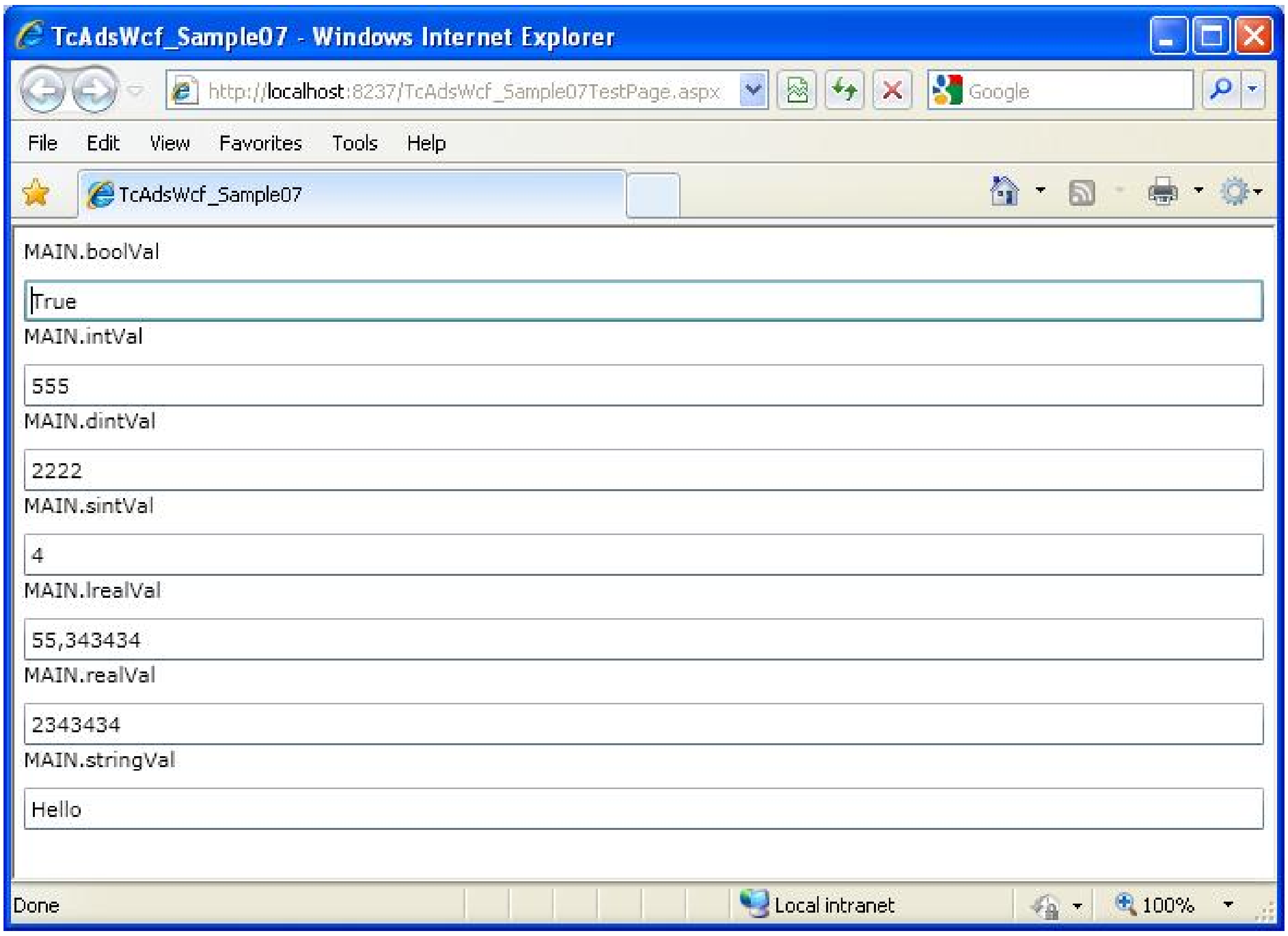
Description

Reading symbol data from the PLC by ADS notification callback on symbol change.
The TwinCAT.Ads.SL DLL (Silverlight) is used to read the data from the recieved byte array in an easy to handle way.

While using the WCF service within a Silverlight application, all methods of the used context interface are appended with the postfix Async.
And there is a extra callback method for each method with the postfix Completed too.

Example: Read1 -> Read1Async, Read1Completed

This is important because silverlight allows asynchronous communication only. A read action by the use of the Read1 method is requested by the use of the Read1Async method and the response will raise the Read1Completed callback.


TcAdsWcf_Sample07 - Windows Internet Explorer
http://localhost:8237/TcAdsWcf_Sample07TestPage.aspx
Google
File Edit View Favorites Tools Help
TcAdsWcf_Sample07
Done
Local intranet
100%

MAIN.boolVal

True

MAIN.intVal

555

MAIN.dintVal

2222

MAIN.sintVal

4

MAIN.lrealVal

55,343434

MAIN.realVal

2343434

MAIN.stringVal

Hello

Silverlight 4 application

```
using System;
using System.ServiceModel;
using System.ServiceModel.Channels;
using System.Windows.Controls;
using TcAdsWcf_Sample07.TwinCAT.Ads.Wcf;
using TwinCAT.Ads.SL;

namespace TcAdsWcf_Sample07
{
    public partial class MainPage : UserControl
    {
    // If the netid string is empty, the TwinCAT ADS WCF service will use the TwinCAT Runtime on its
 host machine.
    private const string NETID = "";
    private const int PORT = 801;

    private TcAdsServiceDuplexClient wcfClient;
    private int[] handles = new int [ 7 ];

    public MainPage ( )
    {
        InitializeComponent ( );

        // Initialize WCF client.
        wcfClient = new TcAdsServiceDuplexClient(
        new PollingDuplexHttpBinding(
            PollingDuplexHttpSecurityMode.None,
            PollingDuplexMode.MultipleMessagesPerPoll ),
        new EndpointAddress ( "http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService/
UnsecPollingDuplex4" ) );

        // Register asynchronous EventHandler.
        wcfClient.AddDeviceNotification1Completed += new EventHandler<AddDeviceNotification1Complete
dEventArgs> ( wcfClient_AddDeviceNotification1Completed );

        // Register CallbackChannel EventHandler.
        wcfClient.AdsNotificationReceived += new EventHandler<AdsNotificationReceivedEventArgs> ( wc
fClient_AdsNotificationReceived );
```

```
        // Subscribe to the AdsNotificationEvent.
        wcfClient.SubscribeAdsNotificationEventAsync ( NETID, PORT );

        // Add device notifications.
        wcfClient.AddDeviceNotification1Async (
        NETID, PORT, "MAIN.boolVal", 1, AdsTransMode.OnChange, 100, 0, null, 0 );
        wcfClient.AddDeviceNotification1Async (
        NETID, PORT, "MAIN.intVal", 2, AdsTransMode.OnChange, 100, 0, null, 1 );
        wcfClient.AddDeviceNotification1Async (
         NETID, PORT, "MAIN.dintVal", 4, AdsTransMode.OnChange, 100, 0, null, 2 );
        wcfClient.AddDeviceNotification1Async (
        NETID, PORT, "MAIN.sintVal", 1, AdsTransMode.OnChange, 100, 0, null, 3 );
        wcfClient.AddDeviceNotification1Async (
        NETID, PORT, "MAIN.lrealVal", 8, AdsTransMode.OnChange, 100, 0, null, 4 );
        wcfClient.AddDeviceNotification1Async (
        NETID, PORT, "MAIN.realVal", 4, AdsTransMode.OnChange, 100, 0, null, 5 );
        wcfClient.AddDeviceNotification1Async (
        NETID, PORT, "MAIN.stringVal", 50, AdsTransMode.OnChange, 100, 0, null, 6 );
    }

    private void wcfClient_AddDeviceNotification1Completed ( object sender, AddDeviceNotification1Co
mpletedEventArgs e )
    {
        int index = ( int ) e.UserState;
        handles [ index ] = e.Result;
    }

    private void wcfClient_AdsNotificationReceived ( object sender, AdsNotificationReceivedEventArgs
 e )
    {
        // Create AdsStream and AdsBinaryReader for buffer handling.
        AdsStream adsStream = new AdsStream ( e.buffer );
        AdsBinaryReader binReader = new AdsBinaryReader ( adsStream );
        adsStream.Position = e.offset;

        // Identify notification by handle and update GUI.
        if ( e.notificationHandle == handles [ 0 ] )
        {
        textBoxBool.Text = binReader.ReadBoolean ( ).ToString ( );
        }
        else if ( e.notificationHandle == handles [ 1 ] )
        {
        textBoxInt.Text = binReader.ReadInt16 ( ).ToString ( );
        }
        else if ( e.notificationHandle == handles [ 2 ] )
        {
        textBoxDint.Text = binReader.ReadInt32 ( ).ToString ( );
        }
        else if ( e.notificationHandle == handles [ 3 ] )
        {
        textBoxSint.Text = binReader.ReadByte ( ).ToString ( );
        }
        else if ( e.notificationHandle == handles [ 4 ] )
        {
           textBoxLreal.Text = binReader.ReadDouble ( ).ToString ( );
        }
        else if ( e.notificationHandle == handles [ 5 ] )
        {
        textBoxReal.Text = binReader.ReadSingle ( ).ToString ( );
        }
        else if ( e.notificationHandle == handles [ 6 ] )
        {
        textBoxString.Text = binReader.ReadPlcString ( e.length ).ToString ( );
        }
    }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481898251/.zip

6.6.3 Reading and writing string variables

Requirements

Visual Studio 2010

Silverlight 4 Tools

TwinCAT ADS WCF

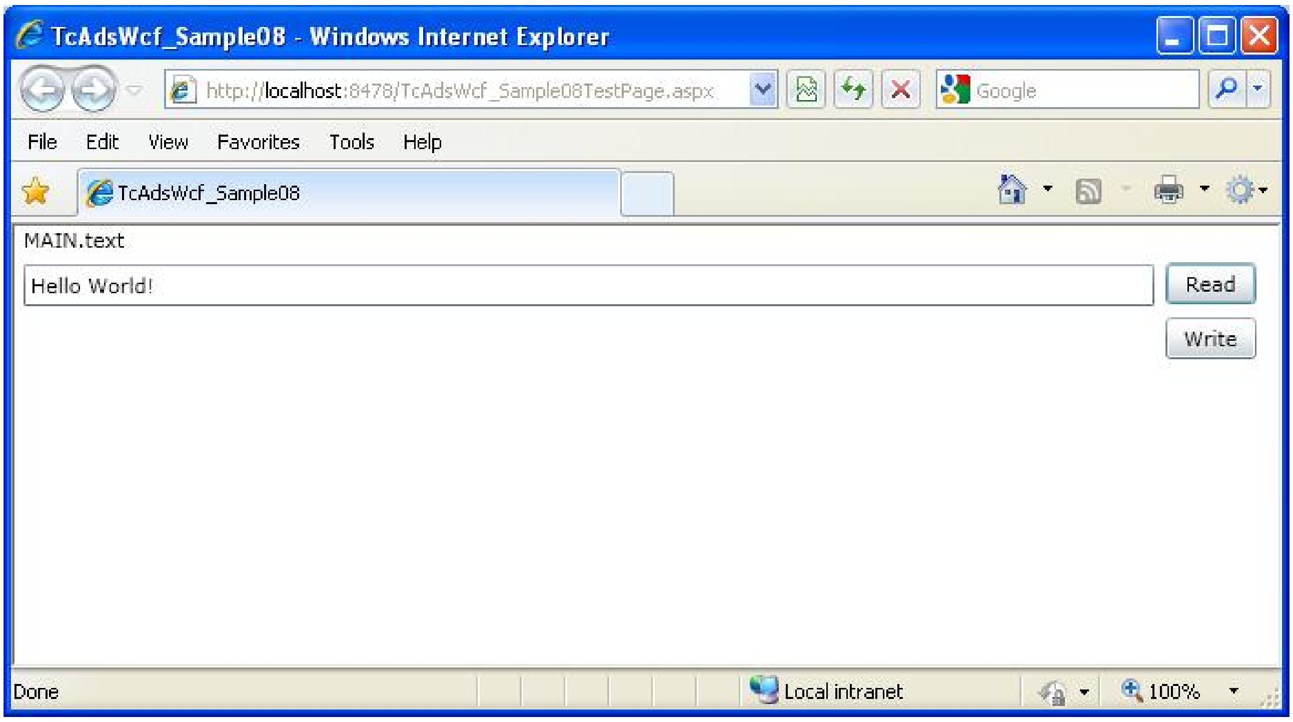
Description

Read a string from a plc variable and display it in the visualization or write a string from the visualization to a plc variable.
The TwinCAT.Ads.SL DLL (Silverlight) is used to read the data from the recieved byte array in an easy to handle way.

While using the WCF service within a Silverlight application, all methods of the used context interface are appended with the postfix Async.
And there is a extra callback method for each method with the postfix Completed too.

Example: Read1 -> Read1Async, Read1Completed

This is important because silverlight allows asynchronous communication only. A read action by the use of the Read1 method is requested by the use of the Read1Async method and the response will raise the Read1Completed callback.


TcAdsWcf_Sample08 - Windows Internet Explorer
http://localhost:8478/TcAdsWcf_Sample08TestPage.aspx
Google
File Edit View Favorites Tools Help
TcAdsWcf_Sample08
Read
Write
Done
Local intranet
100%

MAIN.text

Hello World!

Silverlight 4 application

```
using System;
using System.ServiceModel;
using System.Windows;
using System.Windows.Controls;
using TcAdsWcf_Sample08.TwinCAT.Ads.Wcf;
using TwinCAT.Ads.SL;

namespace TcAdsWcf_Sample08
{
    public partial class MainPage : UserControl
    {
    // If the netid string is empty, the TwinCAT ADS WCF service will use the TwinCAT Runtime on its
 host machine.
    private const string NETID = "";
    private const int PORT = 801;
    private TcAdsServiceSimplexClient wcfClient;

    public MainPage ( )
    {
        InitializeComponent ( );

        wcfClient = new TcAdsServiceSimplexClient (
            new BasicHttpBinding ( ),
            new EndpointAddress ( "http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService/
UnsecBasicHttp" ));

        wcfClient.Read1Completed += new EventHandler<Read1CompletedEventArgs1>(wcfClient_Read1Comple
ted);
    }

    private void wcfClient_Read1Completed(object sender, Read1CompletedEventArgs1 e)
    {
        // Create AdsStream and AdsBinaryReader for buffer handling.
        AdsStream stream = new AdsStream(e.Result);
        AdsBinaryReader binReader = new AdsBinaryReader(stream);
        stream.Position = 0;

        tbText.Text = binReader.ReadPlcString(30);
    }

    private void btnRead_Click ( object sender, RoutedEventArgs e )
    {
        // Read text asynchronous.
        wcfClient.Read1Async ( NETID, PORT, "MAIN.text", 30, null );
    }

    private void btnWrite_Click ( object sender, RoutedEventArgs e )
    {
        // Create a byte array buffer.
        // Length of text +1 because of zero termination.
        byte [ ] buffer = new byte [ tbText.Text.Length + 1 ];

        // Initialize an AdsStream and an AdsBinaryWriter to write the text to the buffer.
        AdsStream stream = new AdsStream ( buffer );
        AdsBinaryWriter binWriter = new AdsBinaryWriter ( stream );
        binWriter.WritePlcString ( tbText.Text, tbText.Text.Length + 1 );

        // Send the buffer.
        wcfClient.Write1Async ( NETID, PORT, "MAIN.text", buffer );
    }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481899659/.zip

6.6.4 Handling SOAP ErrorFaults and specific WCF Exceptions

Requirements

Visual Studio 2010

Silverlight 4 Tools

TwinCAT ADS WCF 1.1.0.1

Silverlight 4 application

```
using System;
using System.Windows;
using System.Windows.Controls;
using TcAdsWcf_Sample10.TwinCAT.Ads.Wcf;
using System.ServiceModel;

namespace TcAdsWcf_Sample10
{
    public partial class MainPage : UserControl
    {
    private TcAdsServiceSimplexClient wcfClient;
    public MainPage ( )
    {
        InitializeComponent ( );

        // Create the TwinCAT ADS WCF client.
        wcfClient = new TcAdsServiceSimplexClient (
            new BasicHttpBinding ( BasicHttpSecurityMode.None ),
            new EndpointAddress ( "http://localhost:8003/TwinCAT/Ads/Wcf/TcAdsService/
UnsecBasicHttp" )
        );

        // Register Read1Completed EventHandler for Read1Async method.
        wcfClient.Read1Completed += new EventHandler<Read1CompletedEventArgs1>(wcfClient_Read1Comple
ted);
    }

    private void Button_Click ( object sender, RoutedEventArgs e )
    {
        // Call Read1Async method with an invalid TwinCAT ADS Symbol name.
        wcfClient.Read1Async ( "", 801, "BAD_VARNAME", 2 );
    }

    private void wcfClient_Read1Completed(object sender, Read1CompletedEventArgs1 e)
    {
        if ( e.Error != null )
        {
        // An error occured while Read1Async was in process.

        if ( e.Error.GetType ( ).Equals (
            typeof ( FaultException<AdsErrorFault> ) ) )
        {
            FaultException<AdsErrorFault> fe
            = e.Error as FaultException<AdsErrorFault>;

            System.Diagnostics.Debug.WriteLine ( "AdsErrorFault: " );
            System.Diagnostics.Debug.WriteLine ( fe.Message );
        }

        if ( e.Error.GetType ( ).Equals (
            typeof ( FaultException<AdsErrorErrorFault> ) ) )
        {
            FaultException<AdsErrorErrorFault> fe
            = e.Error as FaultException<AdsErrorErrorFault>;

            System.Diagnostics.Debug.WriteLine ( "AdsErrorErrorFault: " );
            System.Diagnostics.Debug.WriteLine ( fe.Message );
        }

        if ( e.Error.GetType ( ).Equals (
            typeof ( FaultException<ArgumentErrorFault> ) ) )
        {
            FaultException<ArgumentErrorFault> fe
            = e.Error as FaultException<ArgumentErrorFault>;

            System.Diagnostics.Debug.WriteLine ( "ArgumentErrorFault: " );
            System.Diagnostics.Debug.WriteLine ( fe.Message );
        }

        if ( e.Error.GetType ( ).Equals (
            typeof ( FaultException<UnexpectedErrorFault> ) ) )
        {
            FaultException<UnexpectedErrorFault> fe
            = e.Error as FaultException<UnexpectedErrorFault>;

            System.Diagnostics.Debug.WriteLine ( "UnexpectedErrorFault: " );
            System.Diagnostics.Debug.WriteLine ( fe.Message );
        }

        if ( e.Error.GetType ( ).Equals (
            typeof ( CommunicationException ) ) )
        {
            CommunicationException ce = e.Error as CommunicationException;

            System.Diagnostics.Debug.WriteLine ( "CommunicationException: " );
            System.Diagnostics.Debug.WriteLine ( ce.Message );
        }

        if ( e.Error.GetType ( ).Equals (
           typeof ( TimeoutException ) ) )
        {
            TimeoutException te = e.Error as TimeoutException;

            System.Diagnostics.Debug.WriteLine ( "TimeoutException: " );
            System.Diagnostics.Debug.WriteLine ( te.Message );
        }

        if ( e.Error.GetType ( ).Equals (
           typeof ( Exception ) ) )
        {
            System.Diagnostics.Debug.WriteLine ( "Exception: " );
            System.Diagnostics.Debug.WriteLine ( e.Error.Message );
        }

        return;
        }

        // No error occured.
        System.Diagnostics.Debug.WriteLine ( "Completed without error!" );
    }
    }
}
```

https://infosys.beckhoff.com/content/1033/tcadswcf/Resources/12481901067/.zip

7 ADS Return Codes

Grouping of error codes:
Global error codes: ADS Return Codes [▶ 91]... (0x9811_0000 ...)
Router error codes: ADS Return Codes [▶ 91]... (0x9811_0500 ...)
General ADS errors: ADS Return Codes [▶ 92]... (0x9811_0700 ...)
RTime error codes: ADS Return Codes [▶ 93]... (0x9811_1000 ...)

Global error codes

Hex	Dec	HRESULT	Name	Description
0x0	0	0x98110000	ERR_NOERROR	No error.
0x1	1	0x98110001	ERR_INTERNAL	Internal error.
0x2	2	0x98110002	ERR_NORTIME	No real time.
0x3	3	0x98110003	ERR_ALLOCLOCKEDMEM	Allocation locked – memory error.
0x4	4	0x98110004	ERR_INSERTMAILBOX	Mailbox full – the ADS message could not be sent. Reducing the number of ADS messages per cycle will help.
0x5	5	0x98110005	ERR_WRONGRECEIVEHMSG	Wrong HMSG.
0x6	6	0x98110006	ERR_TARGETPORTNOTFOUND	Target port not found – ADS server is not started or is not reachable.
0x7	7	0x98110007	ERR_TARGETMACHINENOTFOUND	Target computer not found – AMS route was not found.
0x8	8	0x98110008	ERR_UNKNOWNCMDID	Unknown command ID.
0x9	9	0x98110009	ERR_BADTASKID	Invalid task ID.
0xA	10	0x9811000A	ERR_NOIO	No IO.
0xB	11	0x9811000B	ERR_UNKNOWNAMSCMD	Unknown AMS command.
0xC	12	0x9811000C	ERR_WIN32ERROR	Win32 error.
0xD	13	0x9811000D	ERR_PORTNOTCONNECTED	Port not connected.
0xE	14	0x9811000E	ERR_INVALIDAMSLENGTH	Invalid AMS length.
0xF	15	0x9811000F	ERR_INVALIDAMSNETID	Invalid AMS Net ID.
0x10	16	0x98110010	ERR_LOWINSTLEVEL	Installation level is too low –TwinCAT 2 license error.
0x11	17	0x98110011	ERR_NODEBUGINTAVAILABLE	No debugging available.
0x12	18	0x98110012	ERR_PORTDISABLED	Port disabled – TwinCAT system service not started.
0x13	19	0x98110013	ERR_PORTALREADYCONNECTED	Port already connected.
0x14	20	0x98110014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 error.
0x15	21	0x98110015	ERR_AMSSYNC_TIMEOUT	AMS Sync Timeout.
0x16	22	0x98110016	ERR_AMSSYNC_AMSERROR	AMS Sync error.
0x17	23	0x98110017	ERR_AMSSYNC_NOINDEXINMAP	No index map for AMS Sync available.
0x18	24	0x98110018	ERR_INVALIDAMSPORT	Invalid AMS port.
0x19	25	0x98110019	ERR_NOMEMORY	No memory.
0x1A	26	0x9811001A	ERR_TCPSEND	TCP send error.
0x1B	27	0x9811001B	ERR_HOSTUNREACHABLE	Host unreachable.
0x1C	28	0x9811001C	ERR_INVALIDAMSFRAGMENT	Invalid AMS fragment.
0x1D	29	0x9811001D	ERR_TLSSEND	TLS send error – secure ADS connection failed.
0x1E	30	0x9811001E	ERR_ACCESSDENIED	Access denied – secure ADS access denied.

Router error codes

Hex	Dec	HRESULT	Name	Description
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	Locked memory cannot be allocated.
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	The router memory size could not be changed.
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	The mailbox has reached the maximum number of possible messages.
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	The Debug mailbox has reached the maximum number of possible messages.
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	The port type is unknown.
0x505	1285	0x98110505	ROUTERERR_NOTINITIALIZED	The router is not initialized.
0x506	1286	0x98110506	ROUTERERR_PORTALREADYINUSE	The port number is already assigned.

Hex	Dec	HRESULT	Name	Description
0x507	1287	0x98110507	ROUTERERR_NOTREGISTERED	The port is not registered.
0x508	1288	0x98110508	ROUTERERR_NOMOREQUEUES	The maximum number of ports has been reached.
0x509	1289	0x98110509	ROUTERERR_INVALIDPORT	The port is invalid.
0x50A	1290	0x9811050A	ROUTERERR_NOTACTIVATED	The router is not active.
0x50B	1291	0x9811050B	ROUTERERR_FRAGMENTBOXFULL	The mailbox has reached the maximum number for fragmented messages.
0x50C	1292	0x9811050C	ROUTERERR_FRAGMENTTIMEOUT	A fragment timeout has occurred.
0x50D	1293	0x9811050D	ROUTERERR_TOBEREMOVED	The port is removed.

General ADS error codes

Hex	Dec	HRESULT	Name	Description
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	General device error.
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	Service is not supported by the server.
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	Invalid index group.
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	Invalid index offset.
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	Reading or writing not permitted.
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	Parameter size not correct.
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	Invalid data values.
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	Device is not ready to operate.
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	Device is busy.
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	Invalid operating system context. This can result from use of ADS blocks in different tasks. It may be possible to resolve this through multitasking synchronization in the PLC.
0x70A	1802	0x9811070A	ADSERR_DEVICE_NOMEMORY	Insufficient memory.
0x70B	1803	0x9811070B	ADSERR_DEVICE_INVALIDPARM	Invalid parameter values.
0x70C	1804	0x9811070C	ADSERR_DEVICE_NOTFOUND	Not found (files, ...).
0x70D	1805	0x9811070D	ADSERR_DEVICE_SYNTAX	Syntax error in file or command.
0x70E	1806	0x9811070E	ADSERR_DEVICE_INCOMPATIBLE	Objects do not match.
0x70F	1807	0x9811070F	ADSERR_DEVICE_EXISTS	Object already exists.
0x710	1808	0x98110710	ADSERR_DEVICE_SYMBOLNOTFOUND	Symbol not found.
0x711	1809	0x98110711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	Invalid symbol version. This can occur due to an online change. Create a new handle.
0x712	1810	0x98110712	ADSERR_DEVICE_INVALIDSTATE	Device (server) is in invalid state.
0x713	1811	0x98110713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode not supported.
0x714	1812	0x98110714	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification handle is invalid.
0x715	1813	0x98110715	ADSERR_DEVICE_CLIENTUNKNOWN	Notification client not registered.
0x716	1814	0x98110716	ADSERR_DEVICE_NOMOREHDLS	No further handle available.
0x717	1815	0x98110717	ADSERR_DEVICE_INVALIDWATCHSIZE	Notification size too large.
0x718	1816	0x98110718	ADSERR_DEVICE_NOTINIT	Device not initialized.
0x719	1817	0x98110719	ADSERR_DEVICE_TIMEOUT	Device has a timeout.
0x71A	1818	0x9811071A	ADSERR_DEVICE_NOINTERFACE	Interface query failed.
0x71B	1819	0x9811071B	ADSERR_DEVICE_INVALIDINTERFACE	Wrong interface requested.
0x71C	1820	0x9811071C	ADSERR_DEVICE_INVALIDCLSID	Class ID is invalid.
0x71D	1821	0x9811071D	ADSERR_DEVICE_INVALIDOBJID	Object ID is invalid.
0x71E	1822	0x9811071E	ADSERR_DEVICE_PENDING	Request pending.
0x71F	1823	0x9811071F	ADSERR_DEVICE_ABORTED	Request is aborted.
0x720	1824	0x98110720	ADSERR_DEVICE_WARNING	Signal warning.
0x721	1825	0x98110721	ADSERR_DEVICE_INVALIDARRAYIDX	Invalid array index.
0x722	1826	0x98110722	ADSERR_DEVICE_SYMBOLNOTACTIVE	Symbol not active.
0x723	1827	0x98110723	ADSERR_DEVICE_ACCESSDENIED	Access denied.
0x724	1828	0x98110724	ADSERR_DEVICE_LICENSENOTFOUND	Missing license.
0x725	1829	0x98110725	ADSERR_DEVICE_LICENSEEXPIRED	License expired.
0x726	1830	0x98110726	ADSERR_DEVICE_LICENSEEXCEEDED	License exceeded.
0x727	1831	0x98110727	ADSERR_DEVICE_LICENSEINVALID	Invalid license.
0x728	1832	0x98110728	ADSERR_DEVICE_LICENSESYSTEMID	License problem: System ID is invalid.
0x729	1833	0x98110729	ADSERR_DEVICE_LICENSENOTIMELIMIT	License not limited in time.
0x72A	1834	0x9811072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	Licensing problem: time in the future.
0x72B	1835	0x9811072B	ADSERR_DEVICE_LICENSETIMETOLONG	License period too long.

Hex	Dec	HRESULT	Name	Description
0x72C	1836	0x9811072C	ADSERR_DEVICE_EXCEPTION	Exception at system startup.
0x72D	1837	0x9811072D	ADSERR_DEVICE_LICENSEDUPLICATED	License file read twice.
0x72E	1838	0x9811072E	ADSERR_DEVICE_SIGNATUREINVALID	Invalid signature.
0x72F	1839	0x9811072F	ADSERR_DEVICE_CERTIFICATEINVALID	Invalid certificate.
0x730	1840	0x98110730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	Public key not known from OEM.
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	License not valid for this system ID.
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	Demo license prohibited.
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	Invalid function ID.
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	Outside the valid range.
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	Invalid alignment.
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	Invalid platform level.
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	Context – forward to passive level.
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	Context – forward to dispatch level.
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	Context – forward to real time.
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	Client error.
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	Service contains an invalid parameter.
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	Polling list is empty.
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var connection already in use.
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	The called ID is already in use.
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	Timeout has occurred – the remote terminal is not responding in the specified ADS timeout. The route setting of the remote terminal may be configured incorrectly.
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Error in Win32 subsystem.
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	Invalid client timeout value.
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	Port not open.
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	No AMS address.
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Internal error in Ads sync.
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	Hash table overflow.
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	Key not found in the table.
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESYM	No symbols in the cache.
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	Invalid response received.
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	Sync Port is locked.
0x756	1878	0x98110756	ADSERR_CLIENT_REQUESTCANCELLED	The request was cancelled.

RTime error codes

Hex	Dec	HRESULT	Name	Description
0x1000	4096	0x98111000	RTERR_INTERNAL	Internal error in the real-time system.
0x1001	4097	0x98111001	RTERR_BADTIMERPERIODS	Timer value is not valid.
0x1002	4098	0x98111002	RTERR_INVALIDTASKPTR	Task pointer has the invalid value 0 (zero).
0x1003	4099	0x98111003	RTERR_INVALIDSTACKPTR	Stack pointer has the invalid value 0 (zero).
0x1004	4100	0x98111004	RTERR_PRIOEXISTS	The request task priority is already assigned.
0x1005	4101	0x98111005	RTERR_NOMORETCB	No free TCB (Task Control Block) available. The maximum number of TCBs is 64.
0x1006	4102	0x98111006	RTERR_NOMORESEMAS	No free semaphores available. The maximum number of semaphores is 64.
0x1007	4103	0x98111007	RTERR_NOMOREQUEUES	No free space available in the queue. The maximum number of positions in the queue is 64.
0x100D	4109	0x9811100D	RTERR_EXTIRQALREADYDEF	An external synchronization interrupt is already applied.
0x100E	4110	0x9811100E	RTERR_EXTIRQNOTDEF	No external sync interrupt applied.
0x100F	4111	0x9811100F	RTERR_EXTIRQINSTALLFAILED	Application of the external synchronization interrupt has failed.
0x1010	4112	0x98111010	RTERR_IRQLNOTLESSOREQUAL	Call of a service function in the wrong context
0x1017	4119	0x98111017	RTERR_VMXNOTSUPPORTED	Intel VT-x extension is not supported.
0x1018	4120	0x98111018	RTERR_VMXDISABLED	Intel VT-x extension is not enabled in the BIOS.
0x1019	4121	0x98111019	RTERR_VMXCONTROLSMISSING	Missing function in Intel VT-x extension.
0x101A	4122	0x9811101A	RTERR_VMXENABLEFAILS	Activation of Intel VT-x fails.

Specific positive HRESULT Return Codes:

HRESULT	Name	Description
0x0000_0000	S_OK	No error.
0x0000_0001	S_FALSE	No error. Example: successful processing, but with a negative or incomplete result.
0x0000_0203	S_PENDING	No error. Example: successful processing, but no result is available yet.
0x0000_0256	S_WATCHDOG_TIMEOUT	No error. Example: successful processing, but a timeout occurred.

TCP Winsock error codes

Hex	Dec	Name	Description
0x274C	10060	WSAETIMEDOUT	A connection timeout has occurred - error while establishing the connection, because the remote terminal did not respond properly after a certain period of time, or the established connection could not be maintained because the connected host did not respond.
0x274D	10061	WSAECONNREFUSED	Connection refused - no connection could be established because the target computer has explicitly rejected it. This error usually results from an attempt to connect to a service that is inactive on the external host, that is, a service for which no server application is running.
0x2751	10065	WSAEHOSTUNREACH	No route to host - a socket operation referred to an unavailable host.
More Winsock error codes: Win32 error codes			

More Information:
www.beckhoff.com/automation