Kofax Communication Server

TC/SOAP Technical Manual

Version: 10.3.0

Date: 2019-12-13

KOFAX

Legal Notice

Table of Contents

Chapter 1

Preface

Kofax Communication Server, the leading single server solution for business communications, has incorporated SOAP & XML technology in its architecture. This simplifies the integration of different communication services (email, SMS, fax, CTI, voice, etc.) in business processes. For example, when routing a structured XML (eXtensible Markup Language) message to the KCS system during order processing, users can now simultaneously generate an order confirmation via fax or email as well as a delivery confirmation via voice or SMS. The KCS system takes care of all the necessary media conversions (for example processing the SOAP envelope and converting the XML data to voice). With full support for HTML, XSLT and PDF, messages can be formatted to suit different target groups using different media. Connections to new web applications and services can now be established even easier than before.

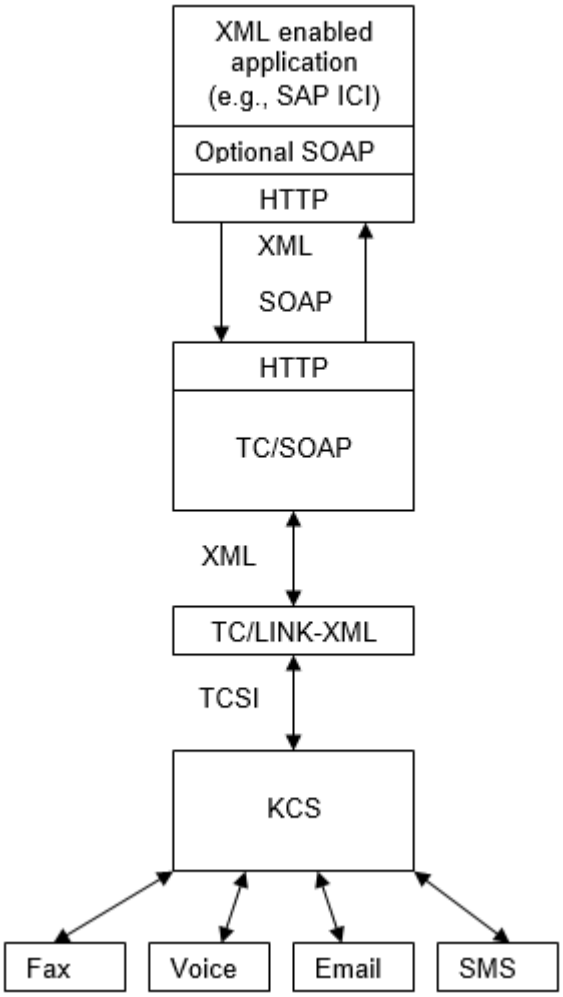
SOAP & XML enable the exchange of meta data and information via the Internet, irrespective of the system deployed. Thus, especially the new technology will become more and more central when it comes to the integration of portal solutions and content management systems (special software for managing and updating comprehensive web offerings). Furthermore, there are new services offered through service providers (xSPs). Kofax Communication Server provides xSPs, a platform for communication solutions made to fit each customer's specific requirements. The xSP solutions enable easy integration of almost all types of media in practically any environment requested by the customers, be that fax distribution from within an application or Unified Messaging over IP.

Important The Kofax Communication Server and its components formerly used the name TOPCALL. Some screen shots and texts in this manual may still use the former name.

Chapter 2

Structure of the Product

Here is small diagram of TC/SOAP:



Chapter 3

Functionality

This section describes the functionality of TC/SOAP.

Scenario: An External Application Sends Plain XML or SOAP Messages in KCS Style

TC/SOAP enables KCS to receive and send SOAP and plain XML messages over the HTTP protocol.

The integrated HTTP Server receives plain XML or SOAP messages. The SOAP messages which are allowed are defined in the WSDL file (tcsoapmsg_w.wsdl or tcsoapmsg.wsdl).

if configured, TC/SOAP will then optionally process the SOAP message and put the XML data into the LINK-XML directory as well as send back the notifications generated by the LINK-XML.

Additionally, TC/SOAP solves the important issue of security. The HTTP server is able to understand HTTPS, which stands for a secure, encrypted connection over the HTTP (SSL).

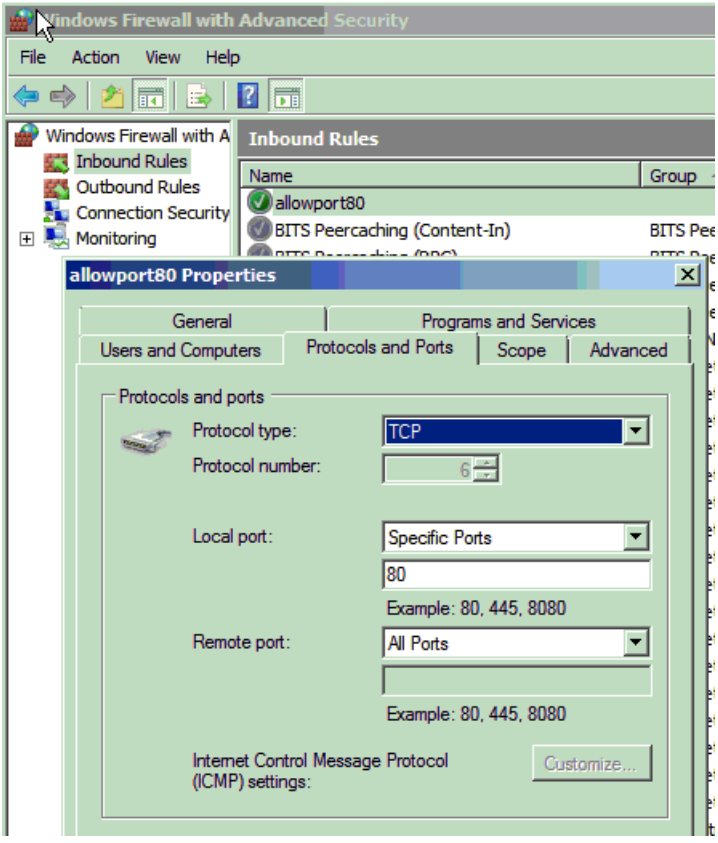
Chapter 4

Prerequisites

Currently, it is necessary for TC/SOAP to install a LINK-XML. No other web server should run on the TCSOAP server.

The TCP ports used for receiving SOAP or HTTP messages must not be blocked by a firewall. With default settings, these are port 80 for normal connections and 443 for secure connections.

On Windows 2008 Server, input via these ports is disabled by default. To enable port 80 (or any other port), use the administrative tool “Windows Firewall with Advanced Security”. Here you can create a new inbound rule that enables the port.



Unicode Support

TC/SOAP can be integrated with UTF-8 based TCOSS systems without character loss. A single message can thus contain texts in different languages, e.g., Russian, Japanese, Chinese, etc.

The following restrictions apply:

- Names of the TC/SOAP and TC/LINK-XML instances must only contain characters of the local code page.
- The temporary directory must only contain characters of the local code page.
- Computer name must only contain characters of the local code page.
- For the optional features, such as “KeepAliveCheck” or “Cancel message”, the configured parameters for logging in to TCOSS (user ID, password, server ID, and server path) must only contain characters of the local code page.

See *TC/LINK-FI Technical Manual* for additional information about Unicode configuration.

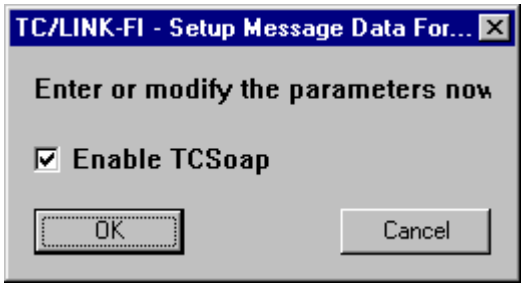
Chapter 5

Installation

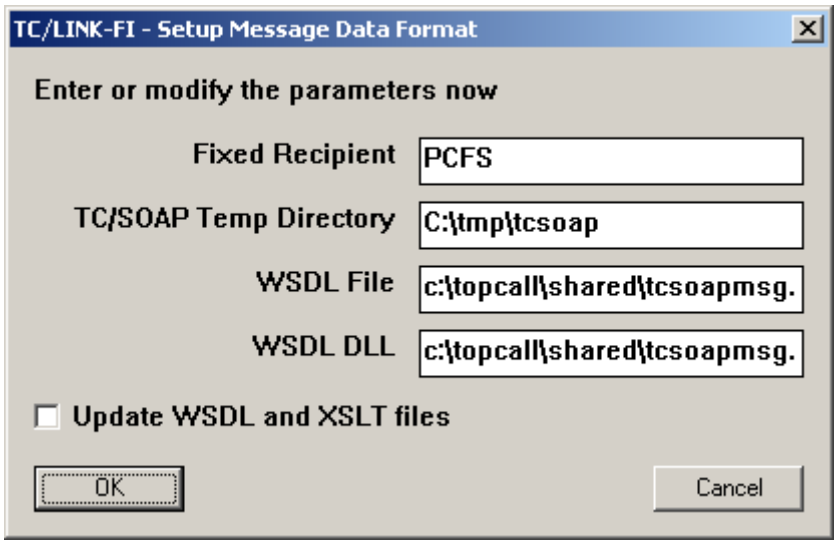
This section describes the installation of TC/SOAP.

Single Instance

TC/SOAP is part of the Kofax Communication Server. It does not have a separate entry in the setup, TC/SOAP is installed as a part of TC/LINK-FI (or LINK-XML). For the specific LINK installation consult the appropriate manual. After the LINK specific setup screens you will be asked if you want to install TC/SOAP:



If you select to enable TCSoap, the following screen is displayed:



Fixed Recipient: In this field, the host name running the external application can be entered. TC/SOAP will send messages to this host only. If no fixed recipient is specified, the receiving host name must be used in the recipient number (e.g. smith@sap_test, whereas “sap_test” would be the host which the message will be sent to). If neither a fixed recipient nor a hostname in the recipient number is specified, the message will not be sent.

TC/SOAP Temp Directory: The TC/SOAP directory where temporary files will be stored.

WSDL File: If you intend to use SOAP, you have to specify the WSDL file which TC/SOAP needs to initialize.
Leave empty if SOAP messages are not used.
Setup installs two WSDL files.

- **tcSoapMsg_w.wsdl**: Contains a WS-I compatible definition of the web services provided by TC/SOAP, using a wrapped document/literal message style.
- **tcSoapMsg.wsdl**: This is the older format, for compatibility with existing installations. It is not WS-I conform, and thus should not be used for new installations.

In Setup, you have to specify the full path name of the WSDL file.

For example, “C:\TOPCALL\SHARED\tcSoapMsg_w.wsdl” for the KCS SOAP Messages (WS-I compatible)

WSDL DLL: If you want to use SOAP, you have to specify the SOAP extension DLL providing the SOAP functions. Leave empty if SOAP messages are not used.

For example, “C:\TOPCALL\SHARED\tcSoapMsg.dll” for the KCS SOAP Messages

Update WSDL and XSLT files: If this checkbox is selected, Setup copies the newest version of tcSoapMsg.wsdl and tcSoapMsg_w.wsdl and the XSLT scripts matching this version to folder C:\TOPCALL\SHARED. Additionally, it configures TC/LINK to use these XSLT scripts for input and output.

You should not activate this checkbox when upgrading an installation where customized WSDL or XSLT files are in use. The WSDL file and scripts delivered with the package may change from version to version.

In any case, copies of the newest WSDL and XSLT files are written to C:\TOPCALL\SHARED:

tcSoapMsg_Newest.WSDL

tcSoapMsg_w_Newest.WSDL

TCXMLSoapOut_Newest.xslt

TCSMLSoapIn_Newest.xslt

On a new installation (without existing WSDL files), this checkbox must be set.

Multiple Instances

If a second TC/SOAP instance is to be installed on the same machine, please proceed as follows:

- Export the TC/SOAP registry hive

- Replace the TC/SOAP string in the registry file with TCSOAP2
- Import the registry file
- Change the registry key “CommandLine” to “C:\TCOSS\TCLP\TCSOAP.EXE TCSOAP2” (or any other directory, you have chosen for the installation)
- Add the “TCSOAP2” string to the “Startup” value under “TOPCALL\Boot”

Both instances can use the same TC/LINK-XML folders and the same temp directory. They cannot listen to the same TCP port, though.

Important Hint for Version Upgrades

In previous versions of the product, SOAP messages had a slightly different structure. Up to TC/LP version 2.24.01, TC/SOAP acting as a SOAP client accepted several attachments defined as XML element <ATT>. From TC/LP version 2.27.06 onwards, the attachments must be indexed <ATT>, <ATT2>, <ATT3> etc. The same is true for recipients of type TO, CC, BCC and AUTH.

When upgrading an installation, please check how the PostMessage parameters are defined in the WSDL file.

If the installation uses a WSDL file without indexed elements, set registry value *SoapProc\AllowMultiples* to 1.

Otherwise, only the first element of a given type is transferred, e.g., only the first attachment.

An example for such a WSDL definition is:

```
    <wsdl:message name="TCPostMessage">
        <wsdl:part name="HEADER" element="xsd1:HEADER"/>
        <wsdl:part name="FROM" element="xsd1:FROM"/>
        <wsdl:part name="TO" element="xsd1:TO"/>
        <wsdl:part name="CC" element="xsd1:CC"/>
        <wsdl:part name="BCC" element="xsd1:BCC"/>
        <wsdl:part name="AUTH" element="xsd1:AUTH"/>
        <wsdl:part name="BODY" element="xsd1:BODY"/>
        <wsdl:part name="ATT" element="xsd1:ATT"/>
    </wsdl:message>
```

Chapter 6

Performance

Expected Performance (based on LINK-XML):

TC/SOAP to KCS:

Single page text document (50 lines of text), sent to KCS user:

0.57 sec/message -> 6350 messages / hour

Large binary attachment (1 MB):

4.4 sec. Transmission time -> 227 Kbytes/second

KCS to TC/SOAP:

Single page fax document (44 Kbytes TCI), via IN-Event to TC/LINK-FI:

0.79 sec/message -> 4500 incoming fax pages / hour

Large binary attachment (1 MB):

7.63 sec. Transmission time -> 131 Kbytes/second

Chapter 7

Conformance to Laws and Directives

Used Standards:

- HTTP 1.1
- SOAP 1.1
- TLS 1.0 / SSL 3.0
- WS-I (with WSDL file tcSoapMsg_w.wsdl)

Chapter 8

Restrictions

This section describes the restrictions.

General Restrictions

The original web service definition used by TCSOAP did not conform to the WS-I standard for web services: It used more than one direct sub element of the SOAP body.
Now, Setup installs a second WSDL file (TCSoapMsg_w.wsdl) that is WS-I standard conform. The original WSDL file (TCSoapMsg.wsdl) is still included for compatibility with existing applications.

Previous versions of the product transmitted sub-elements of the SOAP message in a wrong order. This can be avoided now by setting registry value SOAPPROC\StrictMode to 1.

Currently, it is necessary for TC/SOAP to install a LINK-XML to process the SOAP and XML files. See possible future enhancements for a follow-up project, which will remove this prerequisite.

"HKEY_LOCAL_MACHINE\SOFTWARE\TOPCALL\TCLINKFI\Options\Out\Attachments" must be set to "embedded". This value is set correctly by Setup and shall not be changed.

The XSLT scripts used by TC/LINK-XML must match the SOAP WSDL file used. When installing TC/LINK-XML with TC/SOAP, Setup offers to install the newest XSLT and WSDL files. Note: these XSLT files are different from the standard TC/LINK-XML XSLT files. They support up to 9 file attachments per message (XML elements ATT, ATT2 to ATT9), and up to 9 recipients of every delivery type (TO, TO2 to TO9, CC, CC2 to CC9, BCC, BCC2 to BCC9, AUTH, AUTH2 to AUTH9).

If TC/SOAP is used in SOAP mode, then some of the LINK-XML features are not supported. Some of them can be changed by modifying the WSDL file or the XSLT scripts. Please contact Professional Services for assistance.

- Custom attachments
- Linked attachments
- More than 9 attachments
- More than 9 recipients of the same type
- TEMPLATE parameter
- Alternative addresses

Restrictions for Connecting Two TCOSS Instances via TC/SOAP

In a scenario where two TCOSS instances (TCOSS1, TCOSS2) are connected via TC/SOAP, the following restrictions must be considered.

- Make sure to configure both instances of TC/SOAP and TC/LINK-XML in a compatible way (e.g. both using the same protocol, WSDL file, style sheets). A message that is not accepted by the other TC/SOAP instance may cause endless send retries.
- **Notifications**:

  It makes no sense to send notifications created by TCOSS1 via two TC/SOAP instances to TCOSS2. For, TCOSS, a notification changes the status of a send order, - and there is no matching send order on TCOSS2.

  Therefore, disable automatic polling of notifications by setting registry value

  SOAPPROC\ActiveNotifPolling to 1.

  Notifications will then remain in the NOTIF folder, where they shall be viewed and deleted regularly by an administrator.

  Notifications can be returned for a various reasons, e.g. if the originator address is invalid or the originator is not allowed to send a message.

  Notifications that are not accepted by the other side can even cause loops.

- **Recipient addresses**:

  Make sure that the recipient can be mapped to a user on the receiving TCOSS.
  A message sent from TCOSS1 to “TCFI,UserID” will be delivered to the mailbox of user “UserId” on TCOSS2.

  But if no such user exists, TCOSS2 creates a new message to “TCFI,UserID”; if each TC/SOAP instance has defined the other one as fixed recipient, the message will continue looping between both.

- **DATE and TIME**:

  When using TC/SOAP as a connector between two instances of TC/LINK-FI or TC/LINK-XML, the ambiguous use of the parameters DATE and TIME may lead to confusion: TC/LINK-XML interprets these parameters differently for sending and receiving. In messages from TCOSS, DATE and TIME refer to the time when the message was made available for the link. For messages to KCS, DATE and TIME refer to the requested send time (in the future). This implies that in messages coming from KCS via TC/LINK-FI or XML, these two parameters are always set and contain a date and time that is already over. If this message is transferred to another TC/LINK-FI, the receiving KCS system will send this message as soon as this date and time is reached. If both TCOSS instances have the same system time, there will be no delay.

- **Number of listener threads**:

  The number of listener threads (SOAPPROC\MaxListeners) should be higher than the number of sender threads (SOAPPROC\MaxSender). If you encounter Winsock error 10054 in the trace or event log, increase the number of listeners. This error occurs if an incoming connection cannot be accepted because there are no free listeners and the maximum number of queued connections (SOAPPROC \RcvBackLog) has been exceeded.

Chapter 9

Appendix

This section provides additional information about TC/SOAP.

General

TC/SOAP consists of the following files:

- TCSOAP.EXE: contains the HTTP Server\Client and general message handling. It distinguishes between plain XML and SOAP messages. If it is a SOAP message, it puts the message to the SOAPProc.DLL which executes the RPC function and returns the XML data. TC/SOAP can be configured either to put and get the XML data into the LINK-XML directory or put the data directly to TCOSS (the latter is part of a possible follow-up project).
- SOAPProc.DLL processes the SOAP message and returns the XML data.
- tcSoapMsg.wsdl is the definition of the SOAP message in KCS style.
- tcSoapMsg_w.wsdl is an alternative definition that is WS-I conform (using wrapped document/literal style).
- tcSoapMsg.dll: the SOAP extension DLL which provides the functionality of the functions defined in the WSDL file.

TC/SOAP can be monitored by the KCS Monitor.

External Application -> KCS:

(For the following chapter: TC/SOAP = Server, External Application = Client)

External messages will be received by TC/SOAP over the HTTP protocol. TC/SOAP supports the HTTP GET and POST message type. A HTTP message consists of a HTTP header and the body. A typical plain XML message looks like this:

```
POST HTTP/1.1
Host: tcsoap.company.com
Content-Type: text/xml; charset="utf-8"
Content-Length: 90
<?xml version="1.0" encoding="UTF-8"?>
<MESSAGE xmlns="http://www.topcall.com/XMLSchema/2002/tc/xml">
...<SUBJECT>Simple example</SUBJECT>
...<TO>
......<SERVICE>TOPCALL</SERVICE>
......<NUMBER>MA</NUMBER>
...</TO>
...<TXT>
...this is a test message from TC/link-XML
```

```
...</TXT>
</MESSAGE>
```

First you can see the HTTP header with following header keywords:

Name	Remarks
POST	HTTP message type
Host	Hostname of the connected TC/SOAP instance. This info is important if you want to use TC/SOAP in connection with a MultiTCOSS system. See 11.5.3 for more info
Content-Type	The MIME content type of the body data
Content-Length	The length of the body data

The body consists in this example of a plain XML data block. So no SOAP Processing is necessary and the message will be posted directly into the TCLINK-XML directory.

A typical SOAP message (conforming to “tcSoapMsg.wsdl”) looks like this:

```
POST /PostMessage HTTP/1.1
Host: sap_test
Content-Type: text/xml; charset="utf-8"
Content-Length: 90
SOAPAction: "http://sap_test/PostMessage"
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
           <HEADER>
<SUBJECT>Simple example</SUBJECT>
...     </HEADER>
<FROM>
...<SERVICE>TOPCALL</SERVICE>
<NUMBER>TCTECH</NUMBER>
</FROM>
<TO>
...<SERVICE>TOPCALL</SERVICE>
<NUMBER>MA</NUMBER>
</TO>
<BODY>
<TXT>
...this is a test message from TC/link-XML
</TXT>
......</BODY>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In it you can see an additional HTTP header field called “SOAPAction”, which contains the URI of the SOAP function. The body will be processed by the “SOAPProc.DLL”, and the generated XML data will be put into the TCLINK-XML directory.

If the registry value “ImmediateResponseServer” is set to 1, then TC/SOAP generates an immediate HTTP Response and sends it back to the external application, after the client closes the connection for sending from client to server.

Currently TC/SOAP sends a hard-coded “HTTP 1.1 200 OK”, if plain XML is enabled. In the case of SOAP the response will be generated by the extension DLL and then sent by TC/SOAP.

If you use the SOAP processing, then you need to have set the “ImmediateResponseServer” to 1, because the SOAP protocol requires always a response. At a new installation, configured for SOAP processing, Setup sets the value to 1.

The socket will always be closed by the sending party, i.e. first external application and afterwards TC/SOAP, when it sends the response.

KCS -> External Application:

(For the following chapter: TC/SOAP = Client, External Application = Server)

TC/SOAP polls the TCLINK-XML directory, optionally generates the SOAP envelope, creates the HTTP header and connects to either a fixed recipient (registry value “FixedRecipient”) or the host name found in the TCLINK-XML file.

The messages look very similar to the above one. Here is an example:

```
POST / HTTP/1.1
Host: sap_test
Content-Type: text/xml; charset="utf-8"
Content-Length: 120
SOAPAction: "http://sap_test/PostMessage"
<?xml version="1.0" encoding="utf-8"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<m:HEADER xmlns:m="http://www.topcall.com/tcSoapMsg.xsd1">
<m:SUBJECT>Simple example</m:Subject>
...</m:HEADER>
<m:FROM xmlns:m="http://www.topcall.com/tcSoapMsg.xsd1">
...<m:SERVICE>TOPCALL</m:SERVICE>
<m:NUMBER>TCTECH</m:NUMBER>
</m:FROM>
<m:TO xmlns:m="http://www.topcall.com/tcSoapMsg.xsd1">
...<m:SERVICE>TOPCALL</m:SERVICE>
<m:NUMBER>MA</m:NUMBER>
</m:TO>
...
<m:BODY xmlns:m="http://www.topcall.com/tcSoapMsg.xsd1">
<m:TXT>This is a simple test message</m:TXT>
</m:BODY>
...
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

If the registry value “ImmediateResponseClient” is set to 1, then TC/SOAP awaits an immediate HTTP Response, after TC/SOAP closed the connection for sending from client to server.

Currently, TC/SOAP does not check the HTTP response for error codes. It just removes the HTTP header and acts like described above (External Application -> KCS).

If you use the SOAP processing, then you need to have set the “ImmediateResponseClient” to 1, because the SOAP protocol requires always a response. At a new installation, configured for SOAP processing, Setup sets the value to 1.

The socket will always be closed by the sending party, i.e. first TC/SOAP and afterwards the external application, when it sends the response.

If the distant web-server is not available, the messages will remain in the TC/LINK-FI directories and a configurable number of send retries is done. If all send retries fail, the message is removed from the TC/LINK-FI directory. If TC/LINK-FI is configured to keep messages at “Active-Forwarded” (Topcall\NotifMail = 1), TC/SOAP creates non-delivery notifications for all involved TCOSS send orders, so that the send orders are terminated.

If the send orders were already terminated by TC/LINK-FI (Topcall\NotifMail=0), you can still find the error entries in the TC/SOAP trace file and in the event log.

With default configuration, TC/SOAP does not change the status of the send order after successfully sending the message to the server. You can configure TC/SOAP to set send orders to status “Sent OK” after successful transmission to the server, by setting the TC/LINK-FI registry value Topcall\NotifMail to 1 and the TC/SOAP registry value SOAPPROC\DelNfWhenSent to 1.

Configuration of TCP Ports

TC/SOAP can be configured to use different ports for receiving and sending messages. This is done by setting the registry keys “SOAPPROC\PortToTC” and “SOAPPROC\PortfromTC”. Additionally the registry keys “SOAPPROC\ServerSecureConnection” and “SOAPPROC\ClientSecureConnection” have an impact of the default values. See the following list:

Reception:

If you have the default (80) specified within the PortToTC and ServerSecureConnection=0, TC/SOAP is listening on port 80. If you have the default (80) specified within the PortToTC and ServerSecureConnection=1, TC/SOAP is listening on port 443.

If you have a different port than default specified within the PortToTC, then TCSOAP is listening on the specified port, both for ServerSecureConnection=0 and 1.

Sending:

If you have the default (80) specified within PortFromTC and ClientSecureConnection=0, it will use port 80 for sending. If you have the default (80) specified within PortFromTC and ClientSecureConnection=1, it will use port 443 for sending.

If any other port than 80 is specified for the PortFromTC it will use this port for either secure or unsecured connection.

TC/SOAP API

This section describes TC/SOAP API.

General Info

With this version of TC/SOAP there are currently two SOAP functions available:

- PostMessage
- PostNotif

The standard WSDL file tcSoapMsg.wsdl defines the following parameters for function PostMessage:

```
   <wsdl:message name="TCPostMessage">
       <wsdl:part element="xsd1:HEADER" name="HEADER"/>
       <wsdl:part element="xsd1:FROM" name="FROM"/>
       <wsdl:part element="xsd1:TO" name="TO"/>
       <wsdl:part element="xsd1:TO2" name="TO2"/>
       <wsdl:part element="xsd1:TO3" name="TO3"/>
 … up to …
       <wsdl:part element="xsd1:TO9" name="TO9"/>
       <wsdl:part element="xsd1:CC" name="CC"/>
       <wsdl:part element="xsd1:CC2" name="CC2"/>
       <wsdl:part element="xsd1:CC3" name="CC3"/>
… up to …
       <wsdl:part element="xsd1:CC9" name="CC9"/>
       <wsdl:part element="xsd1:BCC" name="BCC"/>
       <wsdl:part element="xsd1:BCC2" name="BCC2"/>
       <wsdl:part element="xsd1:BCC3" name="BCC3"/>
.. up to …
       <wsdl:part element="xsd1:BCC9" name="BCC9"/>
       <wsdl:part element="xsd1:AUTH" name="AUTH"/>
       <wsdl:part element="xsd1:AUTH2" name="AUTH2"/>
       <wsdl:part element="xsd1:AUTH3" name="AUTH3"/>
… up to …
       <wsdl:part element="xsd1:AUTH9" name="AUTH9"/>
       <wsdl:part element="xsd1:BODY" name="BODY"/>
       <wsdl:part element="xsd1:ATT" name="ATT"/>
       <wsdl:part element="xsd1:ATT2" name="ATT2"/>
       <wsdl:part element="xsd1:ATT3" name="ATT3"/>
… up to …
       <wsdl:part element="xsd1:ATT9" name="ATT9"/>
   </wsdl:message>
```

For further details (e.g. sub-elements of these parameters) see the WSDL file.

The standard WSDL file tcSoapMsg.wsdl defines the following parameters for function PostNotif:

```
   <wsdl:message name="TCPostNotif">
       <wsdl:part element="xsd1:HEADER" name="HEADER"/>
       <wsdl:part element="xsd1:FROM" name="FROM"/>
       <wsdl:part element="xsd1:TO" name="TO"/>
       <wsdl:part element="xsd1:NFINFO" name="NFINFO"/>

       <wsdl:part element="xsd1:BODY" name="BODY"/>

       <wsdl:part element="xsd1:ATT" name="ATT"/>

       <wsdl:part element="xsd1:ATT2" name="ATT2"/>

       <wsdl:part element="xsd1:ATT3" name="ATT3"/>

… up to …

       <wsdl:part element="xsd1:ATT9" name="ATT9"/>
```

```
    </wsdl:message>
```

In the next chapter you will find an example application, which uses a TC/SOAP function to post a message to KCS.

Using TC/SOAP Functions Within .NET

When you install TC/SOAP, you will find a ZIP file (“tcsoapclient.zip”) in the “TOPCALL\SHARED” folder, which contains a sample application, written in C#, which sends a message to KCS via TC/SOAP. Here is the code:

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

namespace tcsoapclient
{...
    class Class1
...   {
......[STAThread]
......static void Main(string[] args)
......{
......sap_test.PortType1Binding request = new sap_test.PortType1Binding();
......sap_test.HEADER requestheader = new sap_test.HEADER();
......sap_test.TO requestto = new sap_test.TO();
......sap_test.FROM requestfrom = new sap_test.FROM();
......sap_test.BODY requestbody = new sap_test.BODY();
......sap_test.RESPONSE response = new sap_test.RESPONSE();
......
......requestto.SERVICE = "TOPCALL";
......requestto.NUMBER = "MA";
......requestfrom.SERVICE = "TOPCALL";
......requestfrom.NUMBER = "MA";
......requestheader.SUBJECT = "Greetings from MA";
......requestbody.TXT = "Hello Test";

response =
request.PostMessage(requestheader,requestfrom,requestto,null,null,null,requestbody,null);

......MessageBox.Show("Response: "+response.INFO);
......}
...   }
    }
```

Here are the necessary steps, If you want to write your own application in .NET:

1. Create a new project (e.g. C# console application)
2. n the “tcsoapmsg.wsdl” file change every occurance of “sap_test” to the computer name, where your TC/SOAP is running.
3. Add a Web Reference in the Solution Reference, by right clicking the Web Reference folder. In the UDDI explorer you can enter the following URL “http://<computername>/tcsoapmsg.wsdl”, whereas the computer name must be the one, where TC/SOAP runs. After the WSDL file is loaded, give it name (in the example above it was “sap_test”) and click on “Add Reference”.

4. After that the objects are created in .NET and you can simply create instances of the objects, like in the example above. E.g. if you want to use “PostMessage”, then you will need following objects:
   - HEADER: It includes the subject
   - FROM: It includes the originator information
   - TO: It includes the recipient information
   - BODY: It includes the body text
   - REQUEST: This will be used for the SOAP request of “PostMessage”
   - RESPONSE: This will store the SOAP response of the “PostMessage” request.
5. Compile and run the application. You should get a message box, which shows you the SOAP response information.

Registry Keys

Location: “HKEY_LOCAL_MACHINE\Software\TOPCALL\TCSOAP\SOAPPROC”

Name	Type	Default	Remarks
ActiveNotifPolling	DWORD	0	Enable Active Notif Polling
AllowMultiples	DWORD	0	This value applies to SOAP sending (acting as a SOAP client). If 1, multiple elements of the same type (e.g. ATT objects) in sequence are allowed. If 0, only 1 element of the same type (e.g. ATT) is allowed and all others are ignored.
AttachmentsLinkedKB	DWORD	1	For incoming SOAP messages, attachments that are bigger than this value (in KB) are passed to TC/LINK-FI as linked attachments. This enhances performance.
CertificatePath	REG_SZ	“C:\TOPCALL\SHARED”	Location of PEM files
CheckContentLength	DWORD	0	0: do not check if number of received bytes matches Content-Length in HTTP header 1: received message is only accepted if number of received bytes matches Content-Length from HTTP header.
CheckWellFormedXML	DWORD	0	Enable XML check
ClientSecureConnection	DWORD	0	Turns on HTTPS connection on client side if set to 1. This means TC/SOAP only accepts HTTP connection but sends out using HTTPS
DelNfWhenSent	DWORD	0	If TC/LINK-FI is configured to keep send orders at status “Active-Forwarded” and this registry value is 1, TC/SOAP creates a delivery notification after successful transmission of a message to the server.
DirectPosting	DWORD	0	Not used yet.

Name	Type	Default	Remarks
DisposeResponse	DWORD	0	For the correct handling of a TCP connection loss, it is recommended to turn on “ImmediateResponseServer” and “ImmediateResponseClient”. If this value is 0, the HTTP responses are stored in the TC/LINK-FI FI_TO_TC folder, which can lead to a message loop because TC/LINK-FI cannot handle them. If this value is 1 (as set after first setup), the HTTP responses are removed.
DynamicRecp	DWORD	0	Set to 1 to enable dynamic recipients as explained in Address Syntax for Dynamic Recipients
FixedRecipient	REG_SZ	“”	Hostname of distant SOAP server. If DynamicRecp is set to 0, then every message is sent to this host. For more info check Address Syntax for Dynamic Recipients
HttpResponseLf	DWORD	0	0: standard line feed (CR LF) in HTTP response 1: non-standard line feed (LF) in HTTP response This key does not apply to SOAP mode.
ImmediateResponseClient	DWORD	0	TC/SOAP awaits a HTTP Response if messages are sent to the external application. Is set to 1 at first installation, if configured for SOAP mode.
ImmediateResponseServer	DWORD	0	TC/SOAP sends a HTTP Response if messages are received from the external application. Is set to 1 at first installation, if configured for SOAP mode.
IPAddress	REG_SZ	“”	If you want to run two TCSOAP instances on one PC, you have to define two IP-Addresses. In this key, each instance has to specify the IP address used.
IPBadList	REG_MULTI_SZ	“”	List of IP-Addresses or domains which messages should be blocked of.
IPGoodList	REG_MULTI_SZ	“”	List of IP-Addresses or domains which messages should be granted of.
LinkSMName	REG_SZ	TCLINKSM	Not used yet.
LinkXMLName	REG_SZ	TCLINKFI	The registry key name of the installed LINK-XML. This is needed to find the XML directories.
MaxListeners	DWORD	10	The maximum number of HTTP Listeners (Max. = 20)
MaxSender	DWORD	10	The maximum number of HTTP Sender (Max. = 20)
MaxSendMessages	DWORD	100	This is the number of send-attempts made per poll cycle. This is needed because otherwise bad send attempts would lead to an endless loop where TC/SOAP would always try to send the messages and the poll cycle would not be used
MaxSendRetries	DWORD	100	Maximum number of send retries. A value of 100 or above means: retry for ever For all other values: if the configured number of retries has been done without success, the message is removed from the folder.
PortFromTC	DWORD	80	This is used to define another HTTP port for sending messages to by TCSOAP (send to).

Name	Type	Default	Remarks
PortToTC	DWORD	80	This is used to define another HTTP port for receiving messages with TCSOAP (listen to).
RcvBackLog	DWORD	4	Size of the connection backlog buffer.
RcvRetryCount	DWORD	10	Configures the count of read retries within the TCP connection.
RcvTimeout	DWORD	2000	This configures the maximum timeout (milliseconds) while waiting for a read event within on RcvRetry Maximum: 1440000 (1 hour)
Routing	REG_MULTI_SZ	“”	Routing table, used for MultiTCOSS instances: “www.tcsoap1.com=TCLINFI www.tcsoap2.com=TCLINKFI2 www.tcsoap3.com=TCLINKFI3”
ServerSecureConnection	DWORD	0	Turns on HTTPS connection on server side, if set to 1. This means TC/SOAP only accepts HTTPS connection but sends out using HTTP
ShortSoapAction	DWORD	0	Defines SOAPAction header field layout when TCSOAP acts as a SOAP client. 0: complete URL,e.g.: SOAPAction: http://servername/PostMessage 1: only function name, e.g.: SOAPAction: PostMessage
SkipRootCertCheck	DWORD	0	0: check validity of web server SSL certificate (must be valid and not self-signed) 1: skip this check
SoapActionPrefix	REG_SZ	“”	Only valid if ShortSoapAction = 0. Defines the URL that shall be used in the SOAPAction header field (i.e. the part between http:// and “/PostMessage”. **Example**: vm-fs-links.local/KCSFunctions Resulting SOAPAction: http://vm-fs-links.local/KCSFunctions/PostMessage If this value is not specified or empty, the SOAPAction value is built from the complete server URL, as specified in fixed or dynamic recipient.
SoapExtension	REG_SZ	“”	Location of the SOAP extension DLL.
SslContextOptions	DWORD	0x03000004	The default value disables SSLv2 and SSLv3 connections, as these protocols are considered insecure. The registry value is a bit mask, individual bit values are defined in the OpenSSL API: https://www.openssl.org/docs/ssl/SSL_CTX_set_options.html
StrictMode	DWORD	0	1: second level XML nodes are re-ordered according to WSDL file definition 0: no such reordering is done
TCPDebug	DWORD	0	Enables TCP trace in the trace file

Name	Type	Default	Remarks
TimeCreated	DWORD	1	The value defines the hours of the file's maximum age. Once, this age is reached, the files will be deleted.
TmpDir	REG_SZ	"C:\TMP \TCSOAP"	Directory for storing temporary files
UpdateWSDL	DWORD	0	Used by Setup only. 1: update WSDL and XSLT files 0: do not update WSDL and XSLT files
WSDLFiles	REG_MULTI_SZ	""	Location of the SOAP WSDL files.

Location: "HKEY_LOCAL_MACHINE\Software\TOPCALL\TCSOAP\TOPCALL"

Name	Type	Default	Remarks
Internal	REG_SZ	""	Used for "Cancel Message" feature. KCS user password (for user specified in registry value TCUserID).
KeepAliveCheck	DWORD	0	Set it to 1 to enable the keep-alive check. Then TC/SOAP checks every minute if TCOSS is still reachable. This removes the problem of filling up the TC/LINK-XML directory while TCOSS is down.
PollCycle	DWORD	30	Poll cycle of checking the LINK-XML (TC_TO_FI) directory.
ServerID	REG_SZ	""	Used for "Cancel Message" feature. ServerID of the KCS system which TCSOAP is connected to.
ServerPath	REG_SZ	""	Used for "Cancel Message" feature. Path to the KCS system which TCSOAP is connected to.
TraceLevel	DWORD	10	Toggles the trace level: 0x01 ... only errors 0x10 ... errors + some functional putput 0x80 ... full output
TCUserID	STRING	""	Used for "Cancel Message" feature. UserID of a KCS user who is allowed to cancel messages.

Under "HKEY_LOCAL_MACHINE\Software\TOPCALL\TCSOAP" you can find the typical KCS specific registry keys, like the trace file settings and the user information where TC/SOAP runs under the TCSRV service.

Scenarios

This section describes various scenarios.

Fault Tolerance (Single TCOSS):

There are 3 possible configurations on one server:

- Single TC/SOAP and multiple LINK-XML (fault tolerant):
  => only possible if multiple LINK-XML polls the same directory because TC/SOAP cannot know if one LINK-XML is not working anymore => possible!

- Multiple TC/SOAP (fault tolerant) and single LINK-XML:
  => only possible if each TC/SOAP listens to different IP-Address. These you have to make visible to the outside (MX records) => possible!
- Multiple TC/SOAP (fault tolerant) and multiple LINK-XML (fault tolerant):
  => combine the first two => possible!

Load Balancing (Single TCOSS):

There are 3 possible configurations on one server:

- Single TC/SOAP and multiple LINK-XML (load balanced):
  => only possible, if multiple LINK-XML polls the same directory, because TC/SOAP cannot know, if one LINK-XML is not working anymore => possible!
- Multiple TC/SOAP (load balanced) and single LINK-XML:
  => Does not make sense, as TC/SOAP already implements multi-listeners (Max.20)
  => single LINK-XML would not be able to handle this amount of messages => no solution!
- Multiple TC/SOAP (load balanced) and multiple LINK-XML (load balanced):
  => multiple TC/SOAP installations on one server require different IP-Addresses or Ports. These must be made visible to the outside (MX records & port-forwarding). But it makes more sense to install only multiple LINK-XML, as TC/SOAP already implements Multi-Listeners (Max.20). => possible!

MultiTCOSS:

There are 3 possible configurations on one Server:

- Single TC/SOAP and multiple LINK-XML (multi TCOSS):
  => Routing from receiving Hostname (HTTP Headerfield “Host:”) to LINK-XML necessary => distant application must use different host names for each TCOSS instance => REG_MULTI_SZ “Routing” used for routing => possible!
- Multiple TC/SOAP (multi TCOSS) and single LINK-XML:
  => not possible, as you need one LINK-XML for each TCOSS instance => no solution!
- Multiple TC/SOAP (multi TCOSS) and multiple LINK-XML (multi TCOSS):
  => multiple TC/SOAP on one server require different IP-Addresses or Ports. These must be made visible to the outside (MX records & port-forwarding) => multiple LINK-XML lead to different instances => no problem, if every TC/SOAP instance leads to one LINK-XML => otherwise you must use the REG_MULTI_SZ “Routing” => possible!

Security

This section describes the security.

HTTPS Support

TC/SOAP implements the “HTTP over SSL” protocol (HTTPS). This is done by using the OpenSSL library. If you want to use SSL, you have to install an SSL certificate, provided by a Certification Authority (CA), and to configure TC/SOAP to use SSL (registry values *ServerSecureConnection* and *ClientSecureConnection* below *SOAPPROC*). To send and receive with SSL, both registry values have to set to 1.

Enabling SSL can impact the ports used for connection. See *Configuration of TCP Ports*.

Additionally, it is important that the connected application is also able to understand HTTPS. SSL works as follows:

Client (e.g. ICI)	Server (e.g. TC/SOAP)
Starts SSL handshake	
	Sends server certificate to client, containing the server public key
Optionally verifies the server certificate	
Sends client certificate to server, containing the client public key	
	Optionally verifies the client certificate
	SSL handshake finished

After the handshake both client and server send their data encrypted with the corresponding public key so that only the owner of the private key can decrypt it.

Therefore, a certificate and private key is needed to use SSL. There are two possibilities: Either you use the self-signed test certificate included in the link installation or use any others from a CA (=certificate authority).

To use the self-signed certificate delivered with TC/SOAP, rename the files PRIVATE_SELFSIGNED.PEM and CERTIFICATE_SELFSIGNED.PEM in the TOPCALL\SHARED folder to PRIVATE.PEM and CERTIFICATE.PEM. This test certificate is valid until 2027, but has two disadvantages:

- It is self-signed, therefore it will not be accepted if the remote computer tries to verify the certification authority against a list of allowed authorities.
- It contains a hard-coded domain name (tclink-sdd-test-cert.topcall.com). If the remote server compares this domain name with the web domain used by TC/SOAP, it will not accept the certificate.

With default configuration, even TC/SOAP does not accept a self-signed certificate when sending to a web server. To allow TC/SOAP to connect to a server with a self-signed certificate, set registry value *SOAPPROC\SkipRootCertCheck* to 1 (this key is created when TC/SOAP starts).

To request a new certificate from a CA, use the OpenSSL tool: File CREATE.BAT contains a sample command line for OPENSSL.

```
openssl req -config openssl.cnf -new -nodes -keyout private.pem -out request.pem -days
 365
```

This command creates a certificate request file, named REQUEST.PEM and a private key file, named PRIVATE.PEM. You have to enter some data in this procedure, where the most important is the “Common Name”. This should be the fully qualified domain name of the TC/SOAP computer (e.g. "tcsoap.company.com").

You can put the file PRIVATE.PEM immediately into the TOPCALL\SHARED directory. After that you have to contact a CA like Thawte (www.thawte.com) and follow the online instructions. You have to provide some information about the company and the desired certificate. It is sufficient to request a standard server certificate.

For more information you should visit the Thawte homepage and if you are interested in the OpenSSL tool: www.openssl.org.

The format of the requested certificate must be PEM. Place the certificate and key e.g. in the TOPCALL \SHARED directory and rename it to CERTIFICATE.PEM and PRIVATE.PEM. It is recommended to use a trusted CA, like Thawte.

To verify certificates, root certificates are needed. Setup will install the root certificates from Thawte into the directory C:\TOPCALL\SHARED. If you get additional root certificates, just open file ROOTCERTS.PEM and append the text from your new certificates to this one. You can add information lines between the certificates, like in the following example:

```
-----END CERTIFICATE--------
original filename: thawteCp.pem
-----BEGIN CERTIFICATE-----
```

If the verification of a certificate fails, you will find an error code in the trace file and the connection will be closed. You can find a list of the error codes at the end of the document (SSL – Verification Error Codes).

It is not possible to mix HTTP with HTTPS because they use different ports. If you want to have support for both on one machine, you have to install two TC/SOAP instances.

The SSL connection is established via the OpenSSL library which is under Open Source license, so it can be used for free.

Good and Bad IP List

TC/SOAP is able to block IP addresses or to grant connections only from specific IP addresses (using the registry keys “IPGoodList” and “IPBadList”).

Solve Possible Duplication of Messages Sent to KCS

In former versions of the product, it was possible to get duplicated messages on KCS, described in the following scenario:

When an application sends a message to TC/SOAP, then it will be first copied into the TC/LINK-XML directory and afterwards the HTTP response is sent to the application. If the TCP connection breaks, before the response is received complete, the application can assume that the message was not sent successfully and repeats the send attempt.

The current TC/SOAP version solves this issue in the following way:

As soon as TC/SOAP gets the message, it puts it into the TC/LINK-XML directory but with a specific file extension “.tm”, so that the Link will not pick them up. Afterwards TC/SOAP sends back the HTTP response and only if the TCP connection was closed gracefully, the file will be renamed to a valid one, so that the Link will get them. Even if TC/SOAP restarts before this renaming occurs, the message is taken care of: TC/SOAP checks for these specific files in the folders at startup and renames them immediately, as long as these files are not too old. The time is configurable via registry key “SOAPPROC\TimeCreated”

Active Notification Polling

This feature is only available for using plain XML over HTTP and not for SOAP!

It is now possible to actively poll TC/SOAP for notifications for messages sent via TC/SOAP to KCS. To enable this you have to set following new registry key under “TCSOAP\SOAPPROC\”:

Name	Type	Default	Remarks
ActiveNotifPolling	DWORD	0	Enable Active Notif Polling

This works the following way: every time a message is received by TC/SOAP, a unique ID is created and stores it first in the message (correlation info field 4) and second send it back in the response, like following example:

```
HTTP/1.1 200 OK  (=HTTP Header of the Response)
TCSOAP20040514_1055253460 (=HTTP Body of the Response, = UniqueID)
```

If you now want to poll for this specific notification, your application have to create a specific HTTP Get Request:

```
GET /TCSOAP20040514_1055253460 HTTP/1.1 (= GET /uniqueID HTTP/1.1)
Host: sap_test  (= Hostname of the PC, where TC/SOAP runs, not so important)
```

If the TC/SOAP finds the correct notification, it will return it in the body of HTTP Response:

```
HTTP/1.1 200 OK  (=HTTP Header of the Response)
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<NOTIFICATION xmlns="http://www.topcall.com/XMLSchema/2002/tc/xml">
<SUBJECT>Delivery Failure: Simple example</SUBJECT>
<FROM>
```

If the notification is not there yet, you will get a failure:

```
HTTP/1.1 404 Not Found

<html><body>Notification not found!</body></html>
```

Cancel Message Feature (Only Possible with Active Notif Polling Enabled)

If you use the Active Notif Polling feature it is also possible to cancel messages on KCS. This feature is enabled by setting the ActiveNotifPolling registry key to 1. Additionally you have to provide the information about the used KCS system in registry key “TCSOAP\TOPCALL”.

Name	Type	Default	Remarks
ServerID	String	“”	KCS ID
ServerPath	String	“”	KCS Path
TCUserID	String	“”	KCS User, which is used to log in to the KCS system, when searching for the user, e.g. the TCLINK user
Internal	String	“”	The password of the KCS User

Also you have to change a line in the ASYSCONFCCCC file in the system folder on TCOSS. Go to line 20 and change the hexadecimal value at position 4 (default : 00) to 2A. Save it and restart TCOSS. The reason for this change is that otherwise it would not be possible for TC/SOAP to search for the unique ID.

To use the cancel message functionality you have to send a HTTP Get method, which looks like the following:

```
 GET /CANCEL:TCSOAP20040524_1614011936 HTTP/1.1
Host: sap_test
```

Important is the “CANCEL:” sequence, followed by the unique ID. The possible HTTP responses follow:

```
 1. (if message successfully cancelled)
HTTP 100 OK
<html><body>Message cancelled</body></html>

2. (configured user has no rights or cannot log in or no user configured)
HTTP 401 Unauthorized
<html><body>Message not cancelled</body></html>

3. (message not there or already terminated or correl4 not configured)
HTTP 404 Message Not Found
<html><body>Message not cancelled</body></html>

4. (message temporarily locked)
HTTP 405 Message temporarily locked - please retry
<html><body>Message not cancelled</body></html>
```

Check for Well Formed XML Message

It is now possible to check the received XML message (only plan XML possible, no SOAP) if they are well formed. There is no XSD or DTD checking implemented. To activate this feature, you have to set the following new registry key under “TCSOAP\SOAPPROC\”:

Name	Type	Default	Remarks
CheckWellFormedXML	DWORD	0	Enable XML check

If the check fails, you will get an immediate response:

```
HTTP/1.1 400 Bad Request
<html><body>XML file not OK</body></html>
```

Address Syntax for Dynamic Recipients

Prerequisite: Create the following registry value “TCSOAP\SOAPPROC\DynamicRecp” (DWORD) and set it to 1.

If you want to use multiple HTTP recipients for messages out of KCS, you can use following syntax as free address:

```
<channel>#<application>@<host>:<port><webdirectory>
```

Example:

```
4711#Myapplication@MyHost:9999|my|directory|soap
```

This address would then be decoded by TC/SOAP and transformed into a HTTP header + new XML elements:

```
POST /my/directory/soap HTTP/1.1
Host: MyHost:9999
Content-Type: text/xml; charset="utf-8"
Content-Length: 90
.
.


<TO>
 <SERVICE>TOPCALL</SERVICE>
```

```
 <NUMBER>4711#Myapplication@MyHost:9999|my|directory|soap</NUMBER>
 <CHANNEL>4711</CHANNEL>
 <APPLICATIONID>Myapplication</APPLICATIONID>
 <HOST>MyHost</HOST>
 <PORT>9999</PORT>
 <WEB>|my|directory|soap</WEB>
</TO>
```

The reason for the “|” sign in the <webdirectory> is because of the forbidden usage of the “/” sign.

If you have only one host which receives the messages, still the “FixedRecipient” and “PortFromTC” is supported and recommended.

Configuration of TCP Receive Behavior

It is now possible to configure both the maximum waiting time and the count of receive retries. For this there are two new registry values under “TCSOAP\SOAPPROC\”:

Name	Type	Default	Remarks
RcvRetryCount	DWORD	10	This configures the count of read retries within the TCP connection
RcvTimeout	DWORD	2000	This configures the maximum timeout (milliseconds) while waiting for a read event within on RcvRetry Maximum: 1440000 (1 hour)

HTTP Response Behavior

Scenario	HTTP Response Header	HTTP Response Body
TCSOAP in HTTP mode, bad XML code received (XML checking enabled)	HTTP/1.1 400 Bad Request	<html><body>XML file not OK</body></html>
TCSOAP in HTTP mode, SOAP request received	HTTP/1.1 500 Bad Request	<html><body>Format error - seems to be SOAP request</body></html>
TCSOAP in SOAP mode, unknown SOAP function requested	HTTP/1.1 500 Bad Request	<html><body>Unknown Function</body></html>
TCSOAP in SOAP mode, request okay and successfully processed. ActiveNotifPolling disabled	HTTP/1.1 200 OK	<html><body>OK</body></html>
TCSOAP in SOAP mode, request okay and successfully processed. ActiveNotifPolling enabled	HTTP/1.1 200 OK <UNIQUEID>	
Active notif polling, notification found	HTTP/1.1 200 OK	The notification
Active notif polling, notification not found	HTTP/1.1 404 Not Found	<html><body>Notification not found!</body></html>
Successful message cancel request	HTTP/1.1 100 OK	<html><body>Message cancelled</body></html>

Scenario	HTTP Response Header	HTTP Response Body
Message cancel failed, missing rights	HTTP/1.1 401 Unauthorized	<html><body>Message not cancelled</body></html>
Message cancel failed, message not found	HTTP/1.1 401 Message not found	<html><body>Message not cancelled</body></html>
Message cancel failed, message locked	HTTP/1.1 401 Message temporarily locked - please retry	<html><body>Message not cancelled</body></html>
Message cancel failed, other error	HTTP/1.1 401 Other error	<html><body>Message not cancelled</body></html>
TCSOAP in SOAP mode, wrong SOAP request	HTTP/1.1 500 Internal Server Error	SOAP response with fault string
TCSOAP in SOAP mode, plain XML received	HTTP/1.1 500 Internal Server Error	SOAP response with fault string
TCSOAP in SOAP mode, SOAP request okay and successfully processed	HTTP/1.1 200 OK	SOAP response
Application connects to TCSOAP without sending any data	TCSOAP will timeout after 30 seconds and it will close the connection	

SSL – Verification Error Codes

Error code	Description
0	**ok**: the operation was successful.
2	**unable to get issuer certificate**: the issuer certificate could not be found: this occurs if the issuer certificate of a non-trusted certificate cannot be found.
3	**unable to get certificate CRL**: the CRL of a certificate could not be found. Unused.
4	**unable to decrypt certificate's signature**: the certificate signature could not be decrypted. This means that the actual signature value could not be determined rather than it not matching the expected value, this is only meaningful for RSA keys.
5	**unable to decrypt CRL's signature**: the CRL signature could not be decrypted: this means that the actual signature value could not be determined rather than it not matching the expected value. Unused.
6	**unable to decode issuer public key**: the public key in the certificate SubjectPublicKeyInfo could not be read.
7	**certificate signature failure**: the signature of the certificate is invalid.
8	**CRL signature failure**: the signature of the certificate is invalid. Unused.

Error code	Description
9	**certificate is not yet valid**: the certificate is not yet valid: the notBefore date is after the current time.
10	**certificate has expired**: the certificate has expired: that is the notAfter date is before the current time.
11	**CRL is not yet valid**: the CRL is not yet valid. Unused.
12	**CRL has expired**: the CRL has expired. Unused.
13	**format error in certificate's notBefore field**: the certificate notBefore field contains an invalid time.
14	**format error in certificate's notAfter field**: the certificate notAfter field contains an invalid time.
15	**format error in CRL's lastUpdate field**: the CRL lastUpdate field contains an invalid time. Unused.
16	**format error in CRL's nextUpdate field**: the CRL nextUpdate field contains an invalid time. Unused.
17	**out of memory**: an error occurred trying to allocate memory. This should never happen.
18	**self signed certificate**: the passed certificate is self signed and the same certificate cannot be found in the list of trusted certificates.
19	**self signed certificate in certificate chain**: the certificate chain could be built up using the non-trusted certificates but the root could not be found locally.
20	**unable to get local issuer certificate**: the issuer certificate of a locally looked up certificate could not be found. This normally means the list of trusted certificates is not complete.
21	**unable to verify the first certificate**: no signatures could be verified because the chain contains only one certificate and it is not self signed.
22	**certificate chain too long**: the certificate chain length is greater than the supplied maximum depth. Unused.
24	**invalid CA certificate**: a CA certificate is invalid. Either it is not a CA or its extensions are not consistent with the supplied purpose.
25	**path length constraint exceeded**: the basicConstraints path length parameter has been exceeded.
26	unsupported certificate purpose: the supplied certificate cannot be used for the specified purpose.
27	**unsupported certificate purpose**: the supplied certificate cannot be used for the specified purpose.

Error code	Description
28	**certificate rejected**: the root CA is marked to reject the specified purpose.
29	**subject issuer mismatch**: the current candidate issuer certificate was rejected because its subject name did not match the issuer name of the current certificate. Only displayed when the -issuer_checks option is set.
30	**authority and subject key identifier mismatch**: the current candidate issuer certificate was rejected because its subject key identifier was present and did not match the authority key identifier current certificate. Only displayed when the -issuer_checks option is set.
31	**authority and issuer serial number mismatch**: the current candidate issuer certificate was rejected because its issuer name and serial number was present and did not match the authority key identifier of the current certificate. Only displayed when the -issuer_checks option is set.
32	**key usage does not include certificate signing**: the current candidate issuer certificate was rejected because its keyUsage extension does not permit certificate signing.
50	**application verification failure**: an application specific error. Unused.

Glossary

ICI – “Integrated Communication Interface”. The ICI is specified by means of SOAP with SOAP-HTTP binding and connects the SAP CRM (“Customer Relationship Management”) with TC/SOAP.

WSDL – “Web Service Definition Language”. WSDL is a document written in XML. The document describes a Web service. It specifies the location of the service and the operations (or methods) the service exposes