Kofax Communication Server

TC/LINK-SI Technical Manual

Version: 10.3.0

Date: 2019-12-13

KOFAX

Table of Contents

Chapter 1

Preface

The product TC/LINK-SI makes it possible to invoke SAP functions without needing third party software.

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.

About this Document

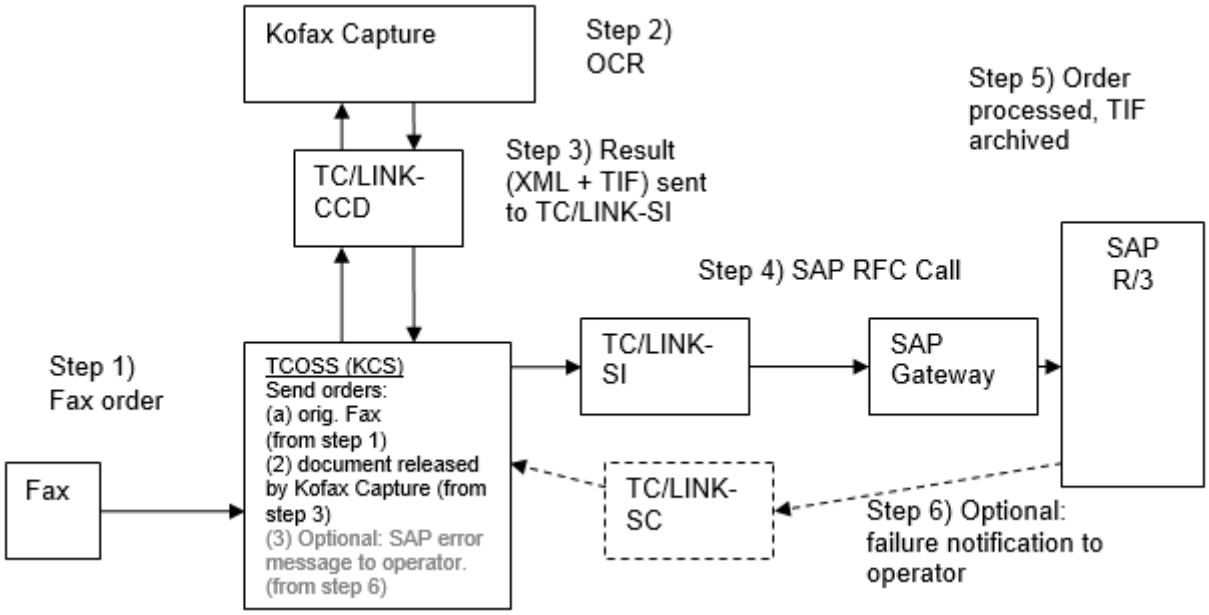
The SAP function call “/TOPCOEB/OEB_POST” which is referred to in the following sections is part of a fictitious sample application.

Benefits, Strengths

- A single link instance can serve several SAP function calls
- One send order leads to 1 SAP function call, using the message content (XML file, other attachments) as input parameters for the function call.
- The function call name is derived from the attached XML file or (as a fallback) from the recipient address.
- 3 files (per function call) tell TC/LINK-SI how to convert the send order into function call parameters:
  *BAPI_X*_DEF.XML file defining the parameters.
  *BAPI_X*_REQUEST.XSLT file defining how to fill the parameters.
  *BAPI_X*_RESPONSE.XSLT file defining how to interpret the response from SAP.
  (*BAPI_X* is just an example here, it stands for a short name of the function call.)
  These files are located in a configurable folder on the link computer. They are created by the technician or by the customer, with help of a tool that is installed with TC/LINK-SI. The files can be edited with any text editor or (more comfortably) with an XML editor.
- Adding support for another function call means adding these 3 files.
- If the function call succeeds, its result can be returned as a sending copy. If the function call fails, its result (or an error message created by TC/LINK-SI) can be forwarded to an operator.

System Overview

The following picture shows TC/LINK-SI in the context of a fictitious application for order processing:



Dotted lines: optional steps

Functionality Overview

Unidirectional operation:

TC/LINK-SI polls send orders from Kofax Communication Server, and issues a SAP RFC call based on the message content.
It acts as a client to SAP. It does not process function calls from SAP.
TC/LINK-SI does not replace TC/LINK-SC. For messaging, you still need TC/LINK-SC.

Message contents:

A send order processed by TC/LINK-SI typically has 1 XML attachment, containing the main parameters for the SAP call. Depending on the function call, additional attachments can be needed, e.g. a TIF attachment with the original document image for order processing.

Special attributes of the KCS send order (e.g. message ID) can also be used as parameters of the function call.

Specifying the function call:

The name of the function call (or a logical short name, see below) is specified in the XML attachment.

As a fallback, this name can also be specified as part of the destination address, e.g.: LinkQueue:FunctionName

Logical short name instead of function call name: The name used here is only allowed to contain characters A-Z, digits 0-9 and the underscore. If the function call exists in a separate SAP namespace,

its name will contain forward slash characters (‘/’). In this case, use a logical name instead of the real function name.

The real function name is specified in the BAPI_X_DEF file.

Error handling:

If TC/LINK-SI encounters a fatal error, the send order on KCS is terminated negatively.

Nonfatal errors (e.g. temporary communication error with SAP) lead to send retries.

In case of fatal errors, the send order can be forwarded to an operator. The operator address can be part of the XML file, thus different operators for different function calls are possible.

Success handling:

If the call to SAP was issued successfully, the send order on KCS is terminated positively.

Optionally, TC/LINK-SI can return the result of the function call (processed via XSLT transformation) as a sending copy.

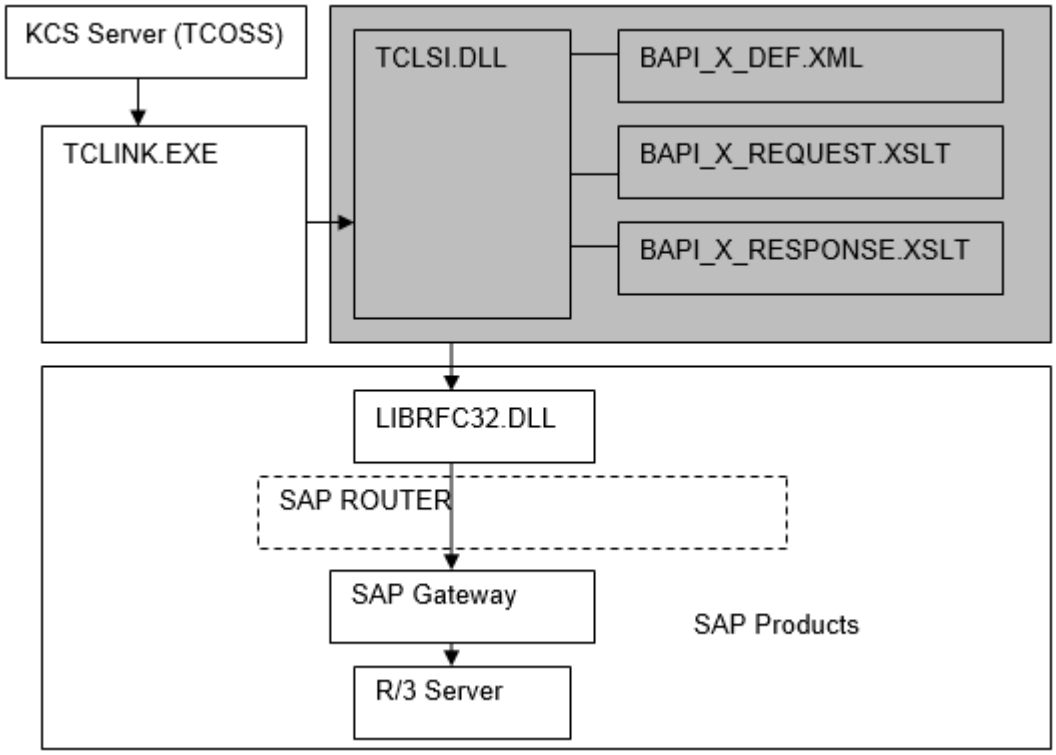
Function call interface to SAP:

Conversion of message content to function call parameters is done via an XSLT transformation.
For every function type, the BAPI_X_DEF.XML file defines the necessary parameters, and the BAPI_X_REQUEST.XSLT file defines how the parameters are filled.

Tools will be provided to create the BAPI_X_DEF.XML file automatically from information exported from SAP.

The XSLT script must be created separately.

Structure of the Product



The shaded part of the drawing contains the new modules:

TCLSI.DLL is the “special link DLL” for TC/LINK-SI. It implements the functions defined for all links. Nevertheless, messages are only retrieved from KCS, never from SAP.

Depending on the content of the send order, TCLSI.DLL looks for the files (BAPI_X_DEF.XML etc.) needed for the SAP function call. In this drawing, a symbolic name (BAPI_X) is used for the function.

TCLSI.DLL uses LIBRFC32.DLL to connect and communicate with the SAP system.

Chapter 2

Features

This section describes the features of TC/LINK-SI.

Invoke SAP Function

TC/LINK-SI picks up a single send orders from Kofax Communication Server. Every send order can invoke one SAP function call. The send order must contain the name of the function call definition file, and parameters for the call.

Typically, most parameters will be in an attached XML file, which was created via OCR. The exact position for each parameter value depends on configuration. The whole process of converting a send order to a function call is based on XML and XSLT conversions.

TC/LINK-SI creates an XML document that holds all information from the message, processes it with an XSLT transformation defined for the function call, and interprets the resulting information as parameters for the SAP function.

This section uses function /TOPCOEB/OEB_POST as an example. This function stores preliminary sales order information (extracted via OCR from a fax) and archives the graphical image of the sales order.

Finding the function name:

The send order must contain information about the function call definition file: either in a text node called BAPI_DEF_NAME below the root element of an XML attachment, or in the recipient address.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <MFundAF>
    <BAPI_DEF_NAME>TOPCOEB_OEB_POST</BAPI_DEF_NAME>
  - <MESSAGE>
    - <ATT ext="txt">
```

Example: XML attachment

If the node BAPI_DEF_NAME does not exist, TC/LINK-SI checks the recipient address.

Example:

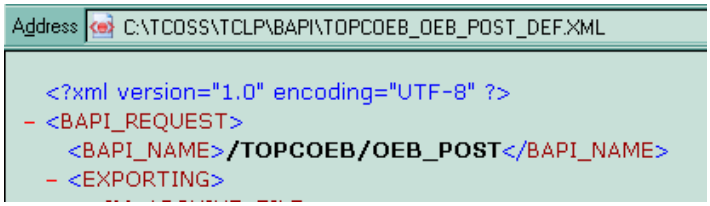
Normalized recipient = “TCLSIQ4:TOPCOEB_OEB_POST”

The address part following the queue name is considered as the base name of the definition file.

The real name of the definition file is <BaseName>_DEF.XML, e.g. TOPCOEB_OEB_POST_DEF.XML.

The file is in the BAPI subfolder of the link directory (configurable).

TC/LINK-SI opens the definition file and reads the real function name from there. A node called BAPI_NAME contains the function name, in this case “/TOPCOEB/OEB_POST”.



Note The definition file name may only contain characters A-Z, 0-9 and the underscore.

For SAP functions that are qualified by a namespace, the separator characters (/) cannot be part of a valid file name. Therefore the base name of the definition file must be different.

Collecting function parameters:

TC/LINK-SI first converts the send order into TC/XML format:

```
<set_entry_ms_mail xmlns="http://www.topcall.com/XMLSchema/2002/tcxl">
  <int_msg_type>49</int_msg_type>
  <ts_ref>BAPI name in recipient address</ts_ref>
  <int_doc_class>8</int_doc_class>
  <time_action>2005-04-07T11:15:48</time_action>
  <ts_file_name>00021379391</ts_file_name>
  <ts_env_name_posted>00021379326</ts_env_name_posted>
  <ts_last_mda_action />
  <int_npag>1</int_npag>
  <int_file_size>1963</int_file_size>
  <time_created>2005-04-07T11:15:44</time_created>
  <ts_document_err />
  <int_susp_dupl>0</int_susp_dupl>
  <ts_tc_msg_id>0146394017118CE0</ts_tc_msg_id>
  <un_content.l_env_cont>
</set_entry_ms_mail>
```

During this step, file attachments are removed and stored as temporary files on the hard disk. Attribute *ext* of the *set_att_obj* node is set to the file extension (converted to uppercase).

```
- <un_content.l_env_cont>
  + <set_header>
    <obj_body_part />
  - <set_att_obj ext=".TIF">
      <ts_comment />
      <ts_appl_id>TCFI004A.TIF</ts_appl_id>
      <ts_long_file_name>TCFI004A.TIF</ts_long_file_name>
      <ts_file_name>TCFI004A.TIF</ts_file_name>
      <int_content_type>1024</int_content_type>
      <ts_tos_folder>C:\TCOSS\TCLP\TMP\TCLINKSAP\XX26BA.tmp</ts_tos_folder>
    </set_att_obj>
  - <set_att_obj ext=".xml">
      <ts_comment />
      <ts_appl_id>example4_input.xml</ts_appl_id>
      <ts_long_file_name>example4_input.xml</ts_long_file_name>
      <ts_file_name>e4_input.xml</ts_file_name>
      <int_content_type>1024</int_content_type>
      <ts_tos_folder>C:\TCOSS\TCLP\TMP\TCLINKSAP\XX26BB.tmp</ts_tos_folder>
    </set_att_obj>
  </un_content.l_env_cont>
```

TC/LINK-SI then adds information from attached XML files. The resulting XML document consists of a ROOT element with 2 child nodes: SendOrders (with message data) and XMLFiles (with data from attached XML files).

```
<?xml version="1.0" encoding="UTF-8" ?>
- <ROOT>
  - <SendOrders>
    - <set_entry_ms_mail xmlns="http://www.topcall.com/XMLSchema/2002/tcxl">
        <int_msg_type>49</int_msg_type>
        <ts_ref>BAPI name in recipient address</ts_ref>
        <int_doc_class>8</int_doc_class>
        <time_action>2005-04-07T11:15:48</time_action>
        <ts_file_name>00021379391</ts_file_name>
        <ts_env_name_posted>00021379326</ts_env_name_posted>
        <ts_last_mda_action />
        <int_npag>1</int_npag>
        <int_file_size>1963</int_file_size>
        <time_created>2005-04-07T11:15:44</time_created>
        <ts_document_err />
        <int_susp_dupl>0</int_susp_dupl>
        <ts_tc_msg_id>0146394017118CE0</ts_tc_msg_id>
      + <un_content.l_env_cont>
      </set_entry_ms_mail>
    </SendOrders>
  - <XMLFiles>
    - <MFundAF>
      - <MESSAGE>
        + <ATT ext="txt">
        </MESSAGE>
      + <Envelope_readattachmentfile>
      </MFundAF>
    </XMLFiles>
  </ROOT>
```

A XSLT file defines the mapping between parts of this XML document and the parameters of the SAP function call. The file is called <BaseName>_REQUEST.XSLT. In this example, the mapping is defined in file TOPCOEB_OEB_POST_REQUEST.XSLT.

The XSLT transformation creates an XML document with the following elements:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <BAPI_REQUEST>
    <BAPI_NAME>/TOPCOEB/OEB_POST</BAPI_NAME>
  + <TCLINKSAP_INSTRUCTIONS>
  + <EXPORTING>
  + <TABLES>
  </BAPI_REQUEST>
```

BAPI_NAME: the name of the function call

EXPORTING: a list of non-tabular parameters for the function call (simple type or structures)

TABLES: a list of tables that the function call expects

TCLINKSAP_INSTRUCTIONS: a node holding additional information used by TC/LINK-SI itself.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <BAPI_REQUEST>
    <BAPI_NAME>/TOPCOEB/OEB_POST</BAPI_NAME>
  + <TCLINKSAP_INSTRUCTIONS>
  - <EXPORTING>
    - <IM_ARCHIVE_FILE>
        <_x002F_TOPCOEB_x002F_OEB_FILENAME>1</_x002F_TOPCOEB_x002F_OEB_FILENAME>
      </IM_ARCHIVE_FILE>
    - <IM_ARCHIVE_FILE_EXT>
        <_x002F_TOPCOEB_x002F_OEB_FILEEXT>TIF</_x002F_TOPCOEB_x002F_OEB_FILEEXT>
      </IM_ARCHIVE_FILE_EXT>
    + <IM_ARCHIV_ID>
    + <IM_ARC_DOC_ID>
    + <IM_AR_OBJECT>
    + <IM_DOCUMENT_TYPE>
    + <IM_OEB_HEAD>
    - <IM_TCC_ID>
        <_x002F_TOPCOEB_x002F_TCC_ID>00021432651</_x002F_TOPCOEB_x002F_TCC_ID>
      </IM_TCC_ID>
    </EXPORTING>
  - <TABLES>
      <T_ARCHIVOBJECT />
    - <T_OEB_ITM>
      - <ITEM>
          <ITM_NUMBER>1</ITM_NUMBER>
          <MATERIAL>P-100</MATERIAL>
          <TARGET_QTY>1</TARGET_QTY>
          <COMP_QUANT>1</COMP_QUANT>
          <TARGET_VAL>2607.6</TARGET_VAL>
        </ITEM>
      </T_OEB_ITM>
    + <T_OEB_PAR>
    </TABLES>
</BAPI_REQUEST>
```

In this picture, the nodes EXPORTING and TABLES are partially expanded.

For table parameters, every table item is below an ITEM node.

Names of parameter and table nodes depend on the SAP function.

If a parameter or table name contains characters that are not allowed in XML node names, these characters must be encoded. The character is replaced by _xHHHH_ (HHHH is the hexadecimal representation of the character). Example:

_x002F_TOPCOEB_x002F_OEB_FILENAME instead of /TCOEB/OEB_FILENAME.

The tool MKBAPIXML2 creates a sample XML file for the request, with already encoded names.

In the following picture, you see how the XSLT script creates the values for item details:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--  Create sample data for BAPI calls   -->
<!--  xmlns:tc="http://www.topcall.com/XMLSchema/2002/tcxl"   -->
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes" />
  - <xsl:template match="/">
    - <BAPI_REQUEST>
        <BAPI_NAME>/TOPCOEB/OEB_POST</BAPI_NAME>
      - <TCLINKSAP_INSTRUCTIONS>
          <NEEDS_COMMIT />
        + <xsl:for-each select="//*[name()='set_att_obj'][@ext='.TIF']">
        </TCLINKSAP_INSTRUCTIONS>
      + <EXPORTING>
      - <TABLES>
          <T_ARCHIVOBJECT />
        - <T_OEB_ITM>
          - <xsl:for-each select="//DOC2ERP_2/Tables/Table
              [@Name='Line_Items']/Rows/Row">
            - <ITEM>
              - <ITM_NUMBER>
                  <xsl:value-of select="position()" />
                </ITM_NUMBER>
              - <MATERIAL>
                  <xsl:value-of select="Cells/Cell[@Name='IDTNR']/@Value" />
                </MATERIAL>
              + <TARGET_QTY>
              + <COMP_QUANT>
              + <TARGET_VAL>
              </ITEM>
```

TLINKSAP_INSTRUCTIONS:

In this example, the node TCLINKSAP_INSTRUCTIONS contains two elements:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--  Create sample data for BAPI calls   -->
<!--  xmlns:tc="http://www.topcall.com/XMLSchema/2002/tcxl"   -->
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes" />
  - <xsl:template match="/">
    - <BAPI_REQUEST>
        <BAPI_NAME>/TOPCOEB/OEB_POST</BAPI_NAME>
      - <TCLINKSAP_INSTRUCTIONS>
          <NEEDS_COMMIT />
        - <xsl:for-each select="//*[name()='set_att_obj'][@ext='.TIF']">
          - <xsl:if test="position()=1">
            - <COPYFILE>
                <DESTINATIONNAME>1.TIF</DESTINATIONNAME>
              - <SOURCENAME>
                  <xsl:value-of select="*[name()='ts_tos_folder']" />
                </SOURCENAME>
                <FOLDER>\\172.16.1.4\topcall</FOLDER>
              </COPYFILE>
            </xsl:if>
          </xsl:for-each>
        </TCLINKSAP_INSTRUCTIONS>
      - <EXPORTING>
        - <IM_ARCHIVE_FILE>

            <_x002F_TOPCOEB_x002F_OEB_FILENAME>1</_x002F_TOPCOEB_x002F_OEB_FILENAME>
          </IM_ARCHIVE_FILE>
        - <IM_ARCHIVE_FILE_EXT>
            <_x002F_TOPCOEB_x002F_OEB_FILEEXT>TIF</_x002F_TOPCOEB_x002F_OEB_FILEEXT>
          </IM_ARCHIVE_FILE_EXT>
        + <IM_ARCHIV_ID>
```

NEEDS_COMMIT: Some SAP functions require a special BAPI_TRANSACTION_COMMIT call after the function all. If TC/LINK-SI finds an X in element NEEDS_COMMIT, it issues this COMMIT call.

COPYFILE: The other node instructs TC/LINK-SI to copy the first TIF attachment of the message to an interface directory. In this example, the resulting file is called 1.TIF and resides in the directory `\\172.16.1.4\topcall`. If the destination file exists already, it will be overwritten.

Function /TOPCOEB/OEB_POST expects the name and extension of the file in the Exporting parameters IM_ARCHIVE_FILE and IM_ARCHIVE_FILE_EXT. The function stores the file in the SAP archive.

Calling the SAP function:

Although TC/LINK-SI uses XML internally, the interface between TC/LINK-SI and SAP is not XML based. It is a simple call interface. All parameters are passed as structures in memory.

To build these structures, TC/LINK-SI must know the exact definition of every parameter, not only its name but also type and size. This information is stored in the file <BaseName>_DEF.XML, e.g. TOPCOEB_OEB_POST_DEF.XML in our example.

TC/LINK-SI fills these structures with the results of the XSLT transformation described above.

Handling of Function Call Results

SAP functions may return complex results.

TC/LINK-SI must interpret the results and decide whether the call was successful or failed.

The TCOSS send order must be updated accordingly. TC/LINK-SI sets the status (success or failure) and a short error code and description.

If the function succeeded, its results can be stored in a back reception document.

In case of errors, TCLINK can forward the send order and the complete set of function call results to an operator.

Interpreting the function results:

TC/LINK-SI converts the return parameters of the function call into an XML document.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <BAPI_REQUEST>
    <BAPI_NAME />
    <EXPORTING />
  - <IMPORTING>
      <EX_ARC_DOC_ID>F3EE445834F6F243B01A5D1C5B9FA267</EX_ARC_DOC_ID>
    - <EX_OLID>
        <_x002F_TOPCOEB_x002F_OLID>0000000072</_x002F_TOPCOEB_x002F_OLID>
      </EX_OLID>
    - <EX_RETURN>
        <TYPE />
        <ID />
        <NUMBER>0</NUMBER>
        <MESSAGE />
        <LOG_NO />
        <LOG_MSG_NO>0</LOG_MSG_NO>
        <MESSAGE_V1 />
        <MESSAGE_V2 />
        <MESSAGE_V3 />
        <MESSAGE_V4 />
        <PARAMETER />
        <ROW>8224</ROW>
        <FIELD />
        <SYSTEM />
      </EX_RETURN>
    </IMPORTING>
  + <TABLES>
  </BAPI_REQUEST>
```

Depending on the SAP function, the results are below the nodes IMPORTING and TABLES.

TC/LINK-SI applies another XSLT transformation to the XML document. The transformation is defined in file <BaseName>_RESPONSE.XSLT. In our example, the file name is TOPCOEB_OEB_POST_RESPONSE.XSLT.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--  Create sample data for BAPI calls  -->
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes" />
  - <xsl:template match="/">
    - <WRAPPER>
      - <TCNotif>
        - <xsl:if test="//IMPORTING/EX_OLID/_x002F_TOPCOEB_x002F_OLID = ''">
            <LastMDANote />
            <LastMDAAction />
            <Status>ERROR</Status>
          </xsl:if>
        - <xsl:if test="//IMPORTING/EX_OLID/_x002F_TOPCOEB_x002F_OLID != ''">
          - <LastMDANote>
              <xsl:value-of
                select="//IMPORTING/EX_OLID/_x002F_TOPCOEB_x002F_OLID" />
            </LastMDANote>
            <LastMDAAction />
            <Status>OK</Status>
          </xsl:if>
        </TCNotif>
      - <TCBackRec>
          <xsl:copy-of select="//IMPORTING" />
        </TCBackRec>
      - <TCBackRecText>
          <xsl:value-of select="concat('the result is:
            ', //IMPORTING/EX_OLID/_x002F_TOPCOEB_x002F_OLID)" />
        </TCBackRecText>
      </WRAPPER>
    </xsl:template>
</xsl:stylesheet>
```

As you can see, this XSLT file is rather simple. Its main purpose is to decide whether the call was successful, and to create a TCNotif node with elements Status, LastMDANote and LastMDAAction.

In this example, the function call is considered as successful if it returns a preliminary sales order ID in the struct EX_OLID.

The Status node can have the value OK or ERROR. LastMDANote and LastMDAAction are optional.

TC/LINK-SI checks the result of the transformation and changes the status of the TCOSS send order accordingly.

TC/LINK-SI can be configured to create a back reception document, which can be forwarded to an operator via a sending copy event. The back reception document can hold a text block and an XML attachment.

Node TCBackRec in the XSLT file defines the content of the backreception's XML attachment. In this example, the IMPORTING node is written to the XML attachment.

Node TCBackRecText in the XSLT file defines the content of the backreception's text block. In this example, the text is "the result is: XXXX", where XXXX is the ID of the preliminary sales order. Please note that carriage return and line feed characters within the TCBackRecText node are ignored. Use "
" instead.

The optional forwarding of error results to an operator is described in the next section.

Reporting Errors to an Operator

If a send order does not result in a successful SAP function call, TC/LINK-SI optionally forwards it to an operator.

Operator:

The operator must exist as a user on Kofax Communication Server.

The name of the operator can be defined globally in TC/LINK-SI configuration (registry value *SAP \Operator*).

But you can also define an operator per message, via the element OPERATOR_NAME in an attached XML file.

The following example shows part of the content of the XML attachment:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <MFundAF>
    <BAPI_DEF_NAME>TOPCOEB_OEB_POST</BAPI_DEF_NAME>
    <OPERATOR_NAME>SAP_OPERATOR</OPERATOR_NAME>
  + <MESSAGE>
  + <Envelope_readattachmentfile>
  </MFundAF>
```

Message to the operator:

The message forwarded to the operator is a copy of the original send order, with an additional XML attachment called SAP_RESULTS.XML.

The original recipient of the send order (containing possibly the function name) is set to inactive.

If the error occurred during the SAP call, the SAP_RESULTS.XML file contains the response from SAP.

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <BAPI_REQUEST>
    <BAPI_NAME />
    <EXPORTING />
  - <IMPORTING>
      <EX_ARC_DOC_ID />
    - <EX_OLID>
        <_x002F_TOPCOEB_x002F_OLID />
      </EX_OLID>
    - <EX_RETURN>
        <TYPE>E</TYPE>
        <ID>/TOPCOEB/OEB</ID>
        <NUMBER>14</NUMBER>
        <MESSAGE>select one row</MESSAGE>
        <LOG_NO />
        <LOG_MSG_NO>0</LOG_MSG_NO>
        <MESSAGE_V1 />
        <MESSAGE_V2 />
        <MESSAGE_V3 />
        <MESSAGE_V4 />
        <PARAMETER />
        <ROW>8224</ROW>
        <FIELD />
        <SYSTEM>ACVCLNT1</SYSTEM>
      </EX_RETURN>
    </IMPORTING>
  + <TABLES>
  </BAPI_REQUEST>
```

If the error occurred before the SAP function call, the file contains error information created by TC/LINK-SI itself:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <DOCUMENT>
    <ERRORTEXT>BAPI_Definition::Call: CallReceive (80004005)</ERRORTEXT>
  </DOCUMENT>
```

The operator can correct the error and send the message to TC/LINK-SI again.

Sending the message:

Messages to the operator are the only type of messages that TC/LINK-SI sends to KCS.

If an error occurs, TC/LINK-SI builds the operator message and stores it as a temporary file in a special folder (defined in registry value *MessagesPath*).

When TC/LINK polls for “messages from SAP”, TC/LINK-SI reads the operator message from the temporary file and posts the message to KCS. The temporary file is locked while it is processed, and it is deleted after the message was sent.

It is possible that several instances of TC/LINK-SI share the same folder for message files.

Chapter 3

Installation

This section describes the installation of TC/LINK-SI.

Installation Steps on the TCOSS Server

This section describes the installation on TCOSS Server,

License

Get a license key for TC/LINK-SI and enter it via the KCS License Maintenance application.

TCOSS Version

Please check if the TCOSS version you are using is supported. Please refer to the TC/LINK manual for information about the minimum required TCOSS version and about restrictions for specific TCOSS version.

You also have to know the KCS server name, the transport type (RPC or Native) and the link type (e.g. TCP/IP) to connect to the KCS server.

KCS Link User

The default name for the link user name is “TCLINK”. From TCOSS 7.22 on this user is automatically created during initial TCOSS installation with password “TCLINK”. Please refer to TC/LINK manual for detailed information.

Create KCS Dependencies

The KCS dependencies can be created automatically when the link starts. Please refer to the TC/LINK manual for detailed information. If you disable this feature, you have to create KCS objects manually. You may not need to create link queue users for all available formats. Choose only those which are required.

Installation Steps on the Link Computer

This section describes the installation on Link computer,

Windows User for TC/LINK

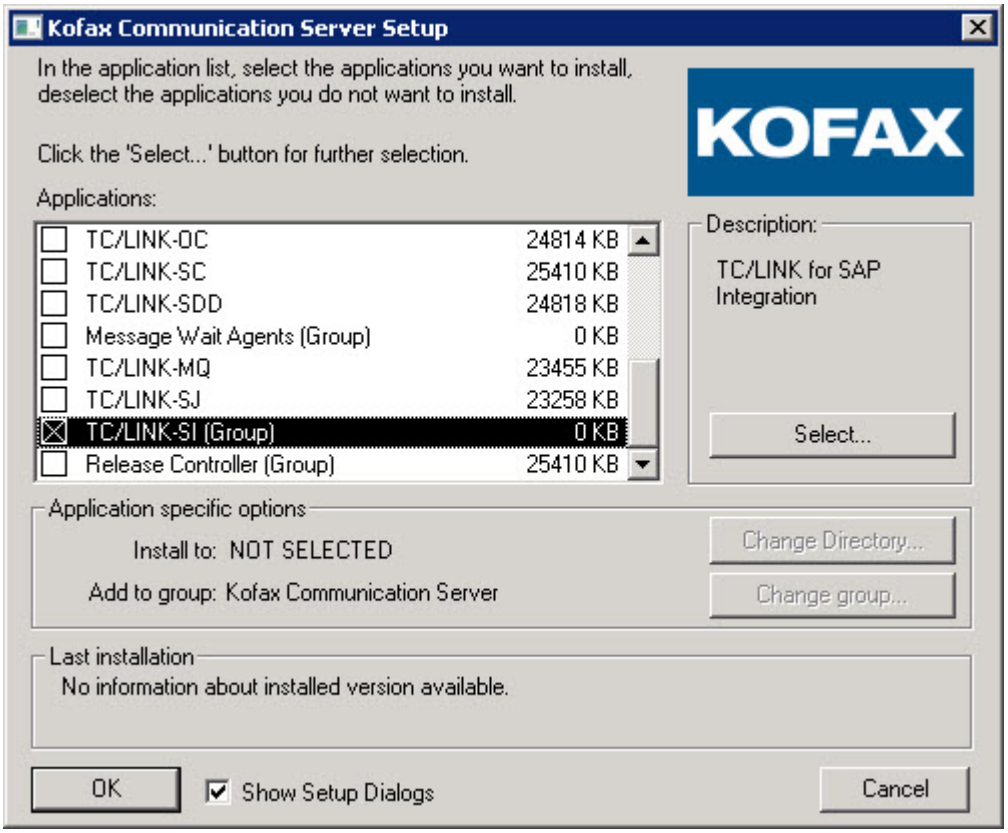
It may be necessary to run TC/LINK-SI with a dedicated user account. For example, the product FaxConnect for Orders uses a directory for passing TIF files to SAP. In this case, TC/LINK-SI should be started with a user account who has write access to this directory. The standard permissions for link users are also needed (member of local Administrators group, Logon As a Batch Job right).

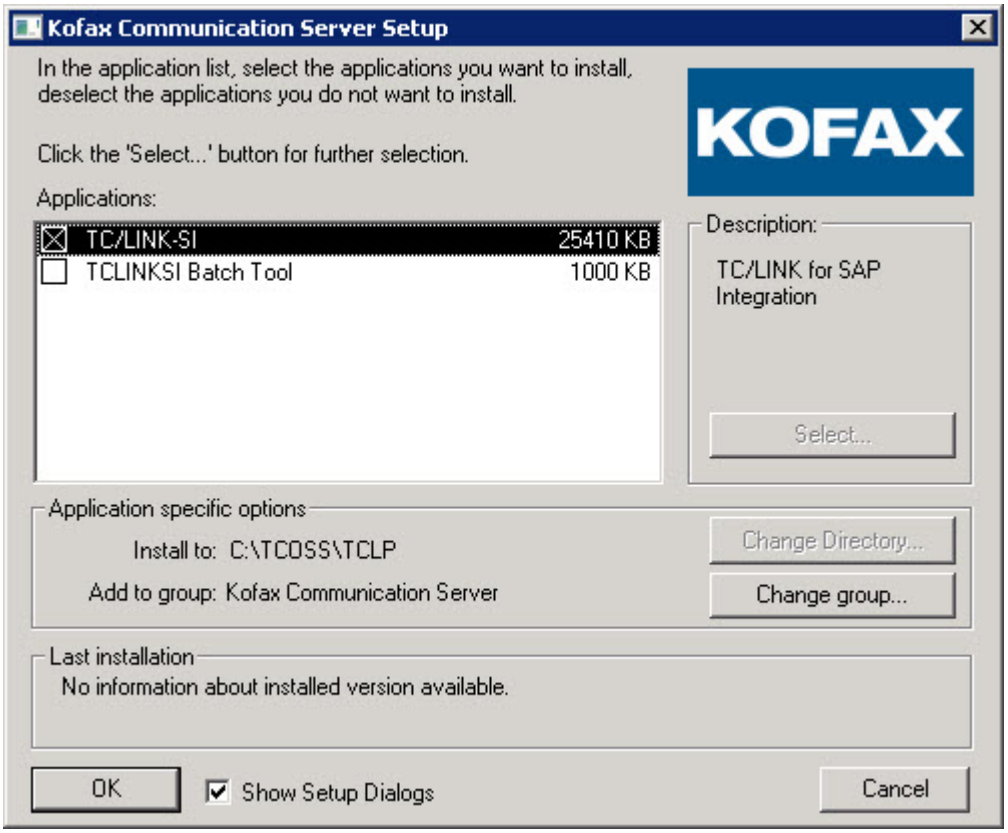
For standard SAP function calls that do not use interface directories, TC/LINK-SI can use the computer account as well.

Run Setup

1. Log in to the Link PC.
2. Run SETUP.EXE from the installation disk.
3. Select the application group TC/LINK-SI (Group).
4. Click **Select**.
5. Select the application TC/LINK-SI and click **OK**.
6. To start setup, press the “Install” button.

7. Please refer to the TC/LINK manual for information about the general part of the installation.





Easy Installation

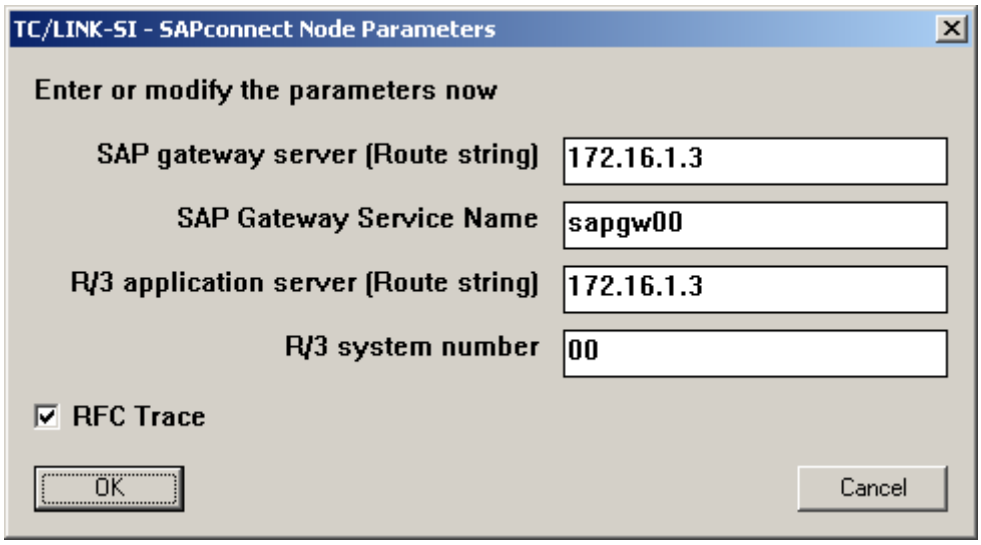
Use this mode for simple new installation and for update. If used for update, settings of the older installation are preserved.

Setup will ask for the SAP connection and login parameters, but will use default values for folder and service names.

Advanced Installation

This mode offers more configuration options (e.g. folder and service names, automatic VPN connection at link startup).

Connection Parameters



SAP gateway server (entry only in SAPRFC.INI file): The IP address (if connected directly, not via SAProuter) or route string (if connected via SAProuter) to the SAP gateway server.

This information must be provided by the SAP System Administrator.

SAP Gateway Service Name (entry only in SAPRFC.INI file): SAP gateway service name or TCP port number where the SAP gateway application listens to. If connected directly this must be the binary TCP port. If connected via SAProuters, this may be the symbolic SAP gateway service name (since SAP symbolic services are evaluated on SAProuter computers.

This information must be provided by SAP System Administrator.

R/3 application server: (entry only in SAPRFC.INI file): The IP address (if connected directly not via SAProuters) or route string (if connected via SAProuters) to the SAP R/3 application server.

This information must be provided by SAP System Administrator.

R/3 system number: the key for R/3 application server's system number (entry in SAPRFC.INI file): System number of the R/3 application server, must be given by SAP System Administrator.

RFC Trace (only in SAPRFC.INI file): If checked, the SAP RFC library creates a trace file for every SAP call (in the folder C:\TCOSS\TCLP). These files contain information about RFC function calls and about the data passed between SAP and TC/LINK-SI. The file names have the prefix "rfc" and the extension "trc". The rest of the file name consists of numbers.

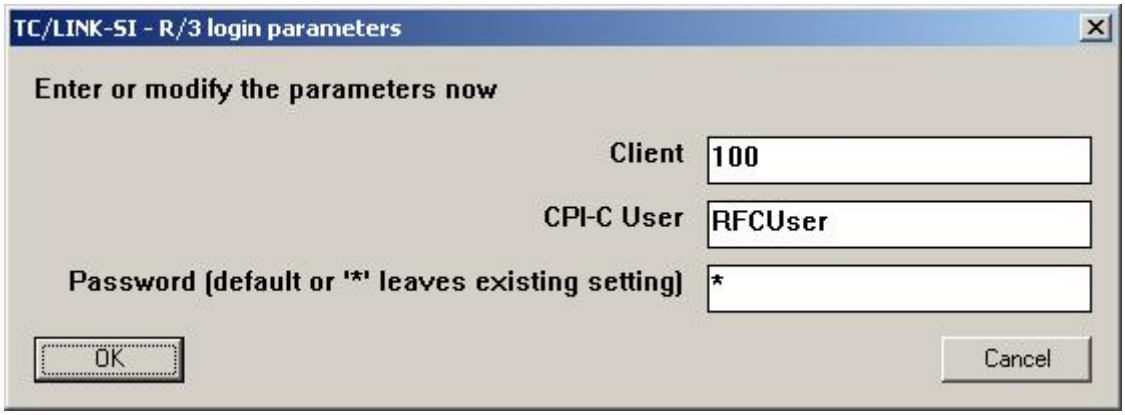
Corresponding SAPRFC.INI file (created by Setup):

```
[CLTCLINKSI]
RFC_TRACE=1
DEST=CLTCLINKSI
TYPE=A
GWHOST=172.16.1.3
```

```
GWSERV=sapgw00
ASHOST=172.16.1.3
SYSNR=00
```

Login Parameters

The next screen asks for login parameters (client ID, user ID, user password):



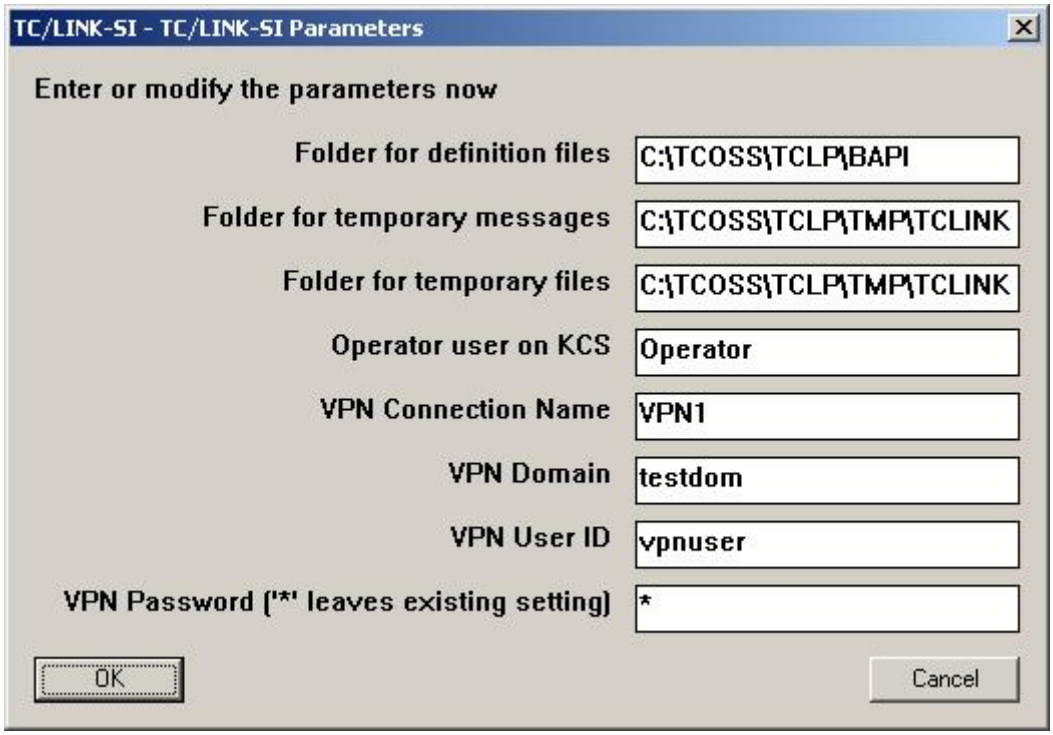
Client (registry: *SAP\RFCClient)* : Client (Mandant) for R/3 application server, where the TC/LINK-SI logs in. Must be provided by SAP System Administrator.

CPI-C User (registry: *SAP\RFCUser*): CPI-C user name on R/3 application server, within the client area specified above. Must be provided by SAP System Administrator.

Password (registry: *SAP\RFCPassword*): Password of this user. Must be provided by SAP System Administrator. Is stored encrypted.

Advanced TC/LINK-SI Parameters

The next screen asks for the special folders used by TC/LINK-SI (only displayed in Advanced Setup):



Folder for definition files (registry: *SAP\BAPIDefsPath*) For every supported SAP function, this folder holds the files <BaseName>_DEF.XML, <BaseName>_REQUEST.XSLT, <BaseName>_RESPONSE.XSLT.

Folder for temporary messages (registry: *SAP\MessagesPath*): Here TC/LINK-SI stores messages that will be sent to the operator.

Make sure to specify a folder that is not automatically cleared at link startup.

Folder for temporary files (registry: *SAP\TempPath*): Here TC/LINK-SI stores temporary files that are deleted after processing a send order (e.g. attachments). This can be the standard TMP folder that is used by the general link routines.

If option “Create dependencies on mail server” is selected during Setup, TCLINK creates the folders for definition files, temporary messages and temporary files right after startup.

Operator user on KCS (registry: *SAP\Operator*): If a function call fails and no operator is specified in the message, this operator user will receive a copy of the message and the results.

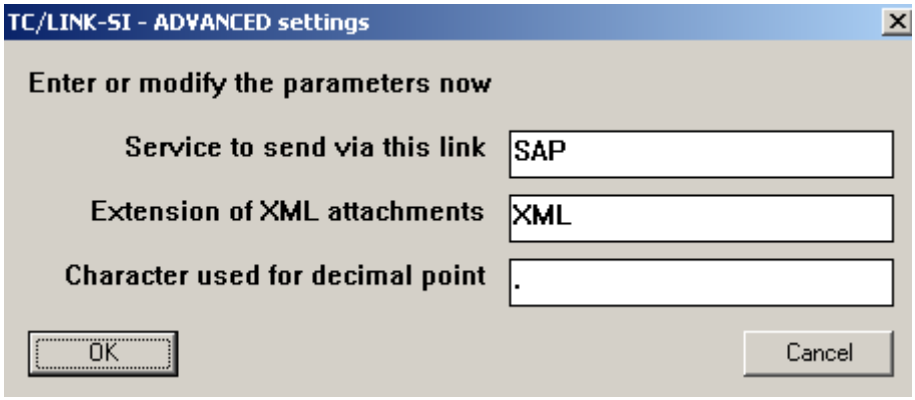
VPN Connection Name (registry: *SAP\RASEntryName*): If the SAP system must be reached via a VPN connection, you can configure TC/LINK-SI to automatically connect. Enter the name of the VPN connection here. If this field is empty, no automatic VPN connect is done.

VPN Domain (registry: *SAP\RASDomain*): Domain for VPN connection

VPN User ID (registry: *SAP\RASUser*): User ID for VPN connection

VPN Password (registry: *SAP\RASPassWord*): Password for VPN connection (stored encrypted)

The next screen is also only available in Advanced Setup:



Service to send via this link (registry: *Setup\ServiceSAP\Name*): Service for SAP function calls, default is SAP.

If option “Create dependencies on TCOSS server” is selected during Setup, TCLINK creates this service after startup.

Extension of XML attachments (registry: *SAP\XMLExtension*): Here you configure which files are treated as XML attachments by TC/LINK-SI. This value is used for all function calls. Therefore make sure that a single extension is used for XML files. Default: XML.

Character used for decimal point (registry: *SAP\DecimalSeparator*): Decimal separator character used in numeric parameters.

Automatic Creation of TCOSS Dependencies

TC/LINK-SI needs several objects on Kofax Communication Server in order to work properly. These KCS objects are called “TCOSS dependencies”.

The only KCS object which must exist (e.g. created via TCfW) before TC/LINK-SI starts, is the TCLINK user who logs in to the TCOSS server. All other KCS objects may be created automatically.

The following dependencies are created automatically during link startup if they do not exist. If they exist with different properties, the link will write an event log entry and terminate. Please check the event log in case of problems.

Description	Default Name	Registry Subkey	More Information
Link queue users	TCLSIQ + image formats		
SAP Service	SAP	Setup\ServiceSAP	Service has free address type and default link queue prefix.

Registry key *TCLINKSI\Setup\CreateDependenciesTopcall* is a flag that tells TCLINK whether it must check and create these TCOSS dependencies. It is set to 1 by SETUP. Whenever TCLINK starts and finds *Setup\CreateDependenciesTopcall* set to 1, it checks and creates the dependencies. TC/LINK does not reset this flag, so by default, dependency check is done at every link start.

Creation of TCOSS dependencies may be disabled by setting this registry value to 0.
Please refer to the TC/LINK manual for detailed information about this feature

Note Common TC/LINK Setup configures several other objects that are created automatically (e.g. default Fax service, templates, cover sheets). Nevertheless, only the objects mentioned above are essential for TC/LINK-SI.

Chapter 4

Configuration

TC/LINK-SI uses the following configuration values, all of them are below subkey SAP:

Value	Type	Default	Description
RFCR3Destination	SZ		Client section in SAPRFC.INI
RFCClient	SZ		SAP client ID
RFCUser	SZ		SAP user ID
RFCPassword	SZ		SAP user password (encrypted)
RFCLanguage	SZ	E	SAP language
CodePage	SZ	1100	SAP codepage
TempPath	SZ	C:\TCOSS\TCLP\TMP\TCLINKSI	Folder used for temporary files (created during message conversion)
Operator	SZ		Name of a KCS user who gets messages that could not be processed.
MessagesPath	SZ	C:\TCOSS\TCLP\TMP\TCLINKSI \MSG	Folder used for messages to the operator. 1 file per message, deleted after message was sent.
BAPIDefsPath	SZ	C:\TCOSS\TCLP\BAPI	Folder that holds function call definitions and XSLT files.
RASEntryName	SZ		Name of remote connection that shall be made automatically.
RASDomain	SZ		Domain for the remote connection
RASUser	SZ		UserId for the remote connection
RASPassword	SZ		Password for the remote connection (encrypted)
XMLExtension	SZ	XML	Only attachments with this extension are interpreted as XML files (i.e. XML content can be used as parameters to the SAP function call)

TC/LINK-SI also uses a new configuration values below subkey General:

Value	Type	Default	Description
BackRecFromMail	DWORD	0	If 1, the result of the function call is stored in a backreception document on KCS.

Additional configuration data is stored in file C:\TCOSS\TCLP\SAPRFC.INI. For a detailed description of the SAPRFC.INI file, please refer to the TC/LINK-SC manual.

Chapter 5

Troubleshooting

This section describes how to troubleshoot errors in TC/LINK-SI.

Trace Output

Registry key *General\Tracelevel* (REG_DWORD) decides how much information is written to the TC/LINK-SI trace file.

Errors: errors are written always to the trace file

Function calls: trace level 50 (decimal) or above

Intermediate XML Files: trace level 128 (decimal) or above

For every SAP function call processed, the link writes the following files to the trace directory:

<Linkname>_REQUEST1_####.XML: TCOSS send order converted to XML format

<Linkname>_REQUEST2_####.XML: TCOSS send order + data from attached XML files

<Linkname>_REQUEST3_####.XML: SAP request after transformation

<Linkname>_RESPONSE1_####.XML: SAP response

In the above description, #### is a placeholder for a file set number (hexadecimal number, starting with 0000, then 0001 etc.). TC/LINK-SI maintains a configurable number of file sets. For every send order, the following files are created:

TC/LINK-SI dumps the file names to the trace file, e.g.:

12/09:56:19.688 (1104) writing temporary file ..\TRACE\TCLINKSI_REQUEST1_0001.XML

The maximum number of file sets is configured in a registry value. The last used file set number is also stored in the registry. When the maximum number is reached, the oldest file set is overwritten.

Configuration:

Registry value name	Type	Default	Description
SAP\ MaxMessageDumps	DWORD	100 decimal	Maximum number of dump file sets
SAP\ IdxMessageDumps	DWORD		Last used file set number

Setup copies such a sample file set to a subdirectory SampleResults below the folder holding the definition files (default: C:\TCOSS\TCLP\BAPI). These sample files show the steps of a successful call of function /TOPCOEB/OEB_POST.

Events

TC/LINK-SI can write the following information to the application event log:

Code	Severity	Description	Parameters
1800	Warning	Error in RFC function %1 (%2): %3	%1: RFC function name %2: SAP function name %3: error description
1801	Warning	Cannot copy file %1 to %2: error %3	%1: source file name %2: destination file name %3: error description
1802	Error	Cannot connect to remote system, RASDial Error %1	%1: error description from RASDial
1803	Warning	Configuration file %1 is missing.	%1: name of configuration file (XML or XSLT file for the SAP function).

Chapter 6

Tools for TC/LINK-SI

This section describes tools for TC/LINK-SI.

MkBAPIXML2.EXE

With this command line tool, you can easily create DEF and XSLT files for SAP functions.

This tool was developed to cure the shortcomings of the old application MkBAPIXML (see below). It connects to SAP by itself, using the configuration values of TC/LINK-SI. This means, it must run on the TC/LINK-SI computer, in order to read the TC/LINK-SI registry.

Call Syntax

```
MKBAPIXML2 LinkName BaseName FunctionName BAPIFolder [Version]
```

LinkName = registry key used by the link, e.g. TCLINKSI

BaseName = base name for output files (A-Z,0-9,_)

FunctionName = name of the RFC function

BAPIFolder = folder for output files

Version = SAP version, currently only 46C or 47 supported, default is 46C

Example:

```
MKBAPIXML2 TCLINKSI TOPCOEB_OEB_POST "/TOPCOEB/OEB_POST" "C:\TCOSS\TCLP\BAPI"
46C
```

MKBAPIXML2 creates the following output files in the output directory:

- <BaseName>_DEF.XML: Definition of the SAP function parameters, including number of decimal digits and maximum number of digits.
- <BaseName>_REQUEST.XML: An XML file containing all parameters for the call. For tables, there is one item per table. Parameter values are all empty.
- <BaseName>_RESPONSE.XML: An XML file containing all parameters that the call can return. For tables, there is one item per table. Parameter values are all empty.

For example:

- TOPCOEB_OEB_POST_DEF.XML
- TOPCOEB_OEB_POST _REQUEST.XML
- TOPCOEB_OEB_POST _RESPONSE.XML

The REQUEST file can be used as a basis for creating the <BaseName>_REQUEST.XSLT transformation file.

The RESPONSE file can be used as a basis for creating the <BaseName>_RESPONSE.XSLT transformation file.

Encoding of parameter names:

All parameter and field names that would be invalid XML node names are encoded.
An invalid character is replaced with _xHHHH_ (HHHH is the hexadecimal representation of the character).

This is done in all 3 result files. When the function call is done, the correct parameter names are transmitted to SAP.

Location:

MKBAPIXML2 is installed into the link directory and it must be started from there (otherwise the program cannot access the SAPRFC.INI file).

MkBAPIXML.EXE

The application MkBAPIXML.EXE is installed into the link directory.

With this tool, you can easily create DEF and XSLT files for SAP functions.
As an input, you need the interface definition of the function, as exported from SAP.

The tool has several disadvantages, and should therefore be used only if MkBAPIXML2.EXE cannot be used.

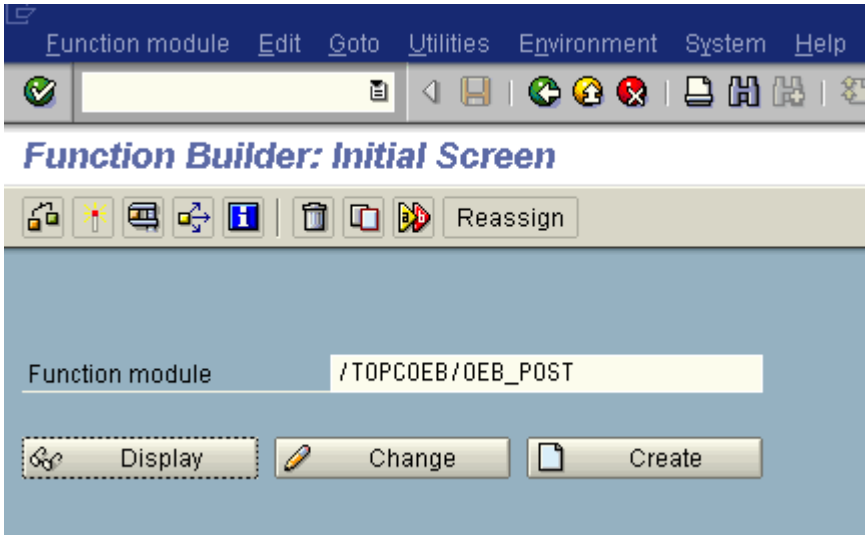
Disadvantages:

- MkBAPIXML cannot export the correct size of numeric parameters.
- It cannot export the number of decimal digits.
- It relies on functionality that is not supported in on a 6.10/6.20/4.7 SAP System, - namely the automatic creation of RFC client code.

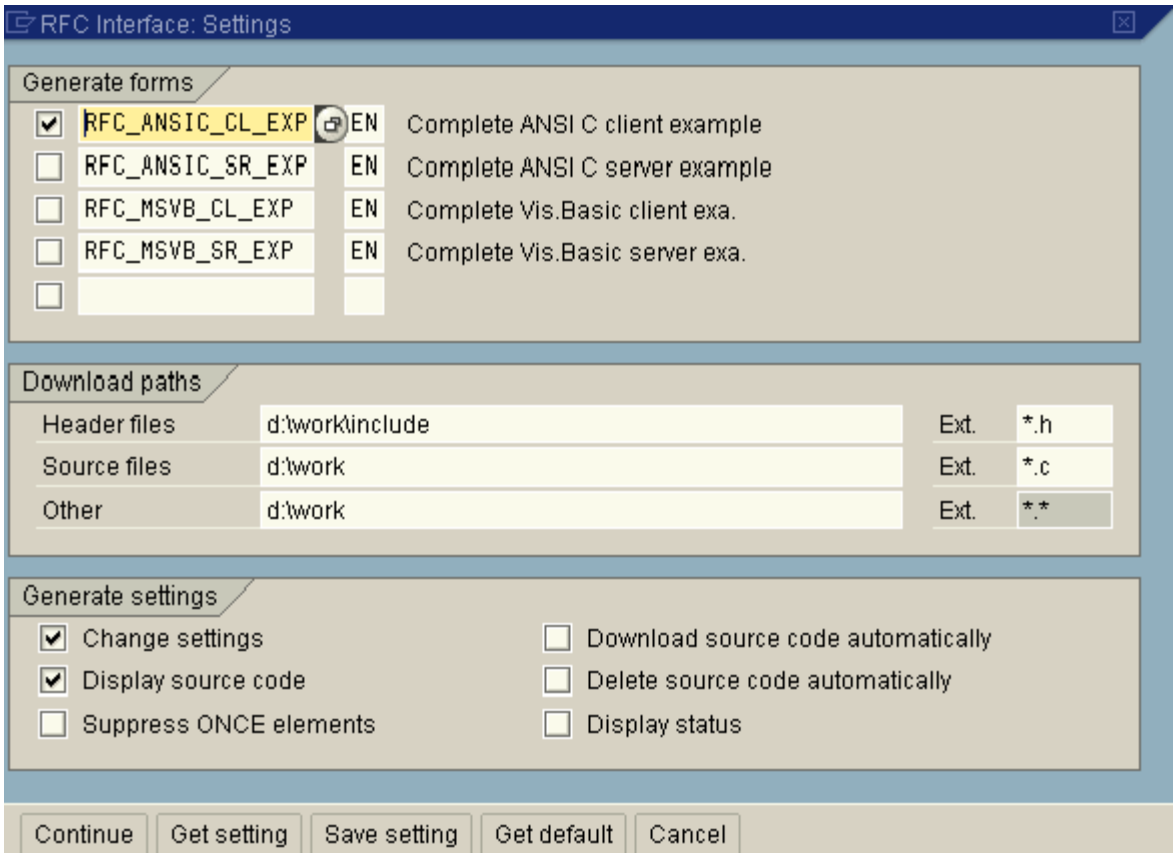
Create input file:

SAPGUI can create automatically a C source file for calling an SAP function.

Choose transaction SE37 (function builder) and enter the function name:

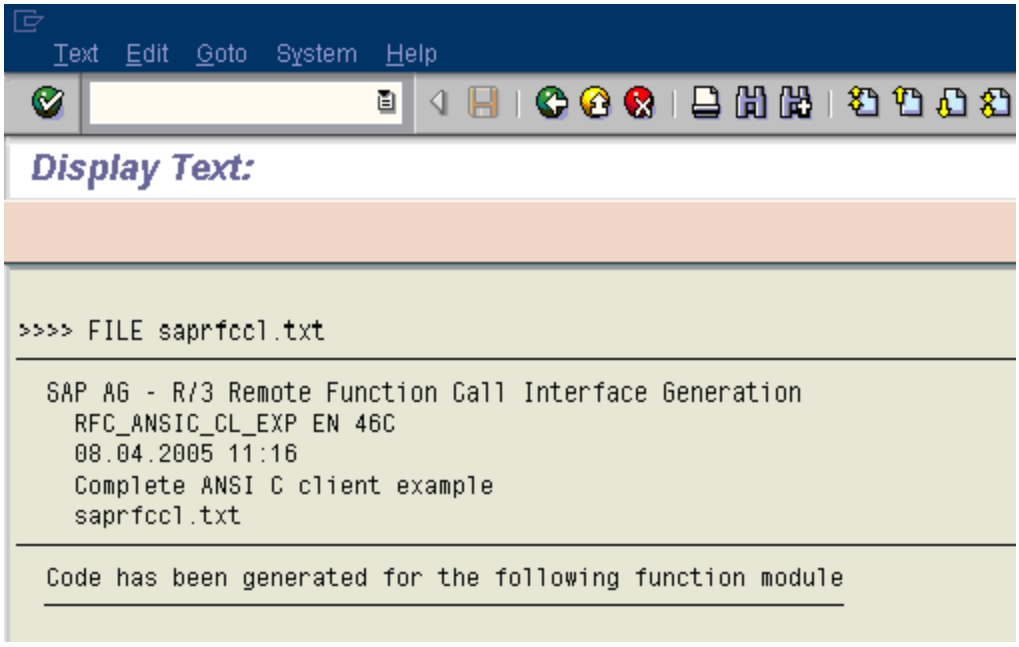


Then choose menu item Utilities|RFC Interface|Generate.



Choose the form “Complete ANSI C client example” and click Continue.

It may take some minutes until the C source code is generated and displayed.



Save the file via menu item System|List|Save|Local File (unconverted).

Command line parameters:

You can then invoke the tool with the following parameters:

Parameter 1: base name (base name for DEF file and XSLT files). Only characters A-Z,0-9 and _ allowed. If the function name contains only allowed characters, you can use the function name as the base name.

Parameter 2: input file (the C file)

Parameter 3: output directory

Example:

MKBAPIXML TOPCOEB_OEB_POST "C:\TEMP\saprfccl1.c" "C:\TCOSS\TCLP\BAPI "

Output:

The tool creates the same set of output files as MKBAPIXML2 does.

Changes:

If the function uses BCD parameters, you must adjust the number of decimals in the resulting DEF file, because the tool stores all BCD numbers as having 0 decimal digits.

Example: the fields EX_RATE_FI and EXCHG_RATE have 5 decimal digits.

You must change their settings from:

```
<EX_RATE_FI length="5" type="2" decimals="0" />
<ADD_VAL_DY length="2" type="6" decimals="0" />
<FIX_VAL_DY length="8" type="1" decimals="0" />
<PMNTTRMS length="4" type="0" decimals="0" />
<PYMT_METH length="1" type="0" decimals="0" />
<ACCNT_ASGN length="2" type="0" decimals="0" />
<EXCHG_RATE length="5" type="2" decimals="0" />
```

to:

```
<EX_RATE_FI length="5" type="2" decimals="5" />
<ADD_VAL_DY length="2" type="6" decimals="0" />
<FIX_VAL_DY length="8" type="1" decimals="0" />
<PMNTTRMS length="4" type="0" decimals="0" />
<PYMT_METH length="1" type="0" decimals="0" />
<ACCNT_ASGN length="2" type="0" decimals="0" />
<EXCHG_RATE length="5" type="2" decimals="5" />
```

TCLSI_BATCH

The TCLINKSI Batch Tool is a standalone application that can be installed via Kofax Communication Server setup. It is also part of the TC/LINK-SI application group in setup.

Once a day, TC/LINK-SI calls a batch of SAP functions, where the result of one call can be input for the next call. Results can be stored as XML, text file (e.g. CSV) or in an ODBC compliant database. What exactly is done depends on configuration.

The program does not access SAP directly, but uses TC/LINK-SI to call the SAP functions and report results back as a sending copy.

For more information, please consult the TCLSI_BATCH manual.

Chapter 7

Restrictions

- TC/LINK-SI supports only those SAP functions that can be called from remote (via Remote Function Calls).
- TC/LINK-SI does not support functions with nested structures as parameters (structure in structure, table in structure).
- Only 1 SAP function per send order is called (+ an optional call of BAPI_TRANSACTION_COMMIT).

Chapter 8

Setup Checklist

TCOSS Server CPU number	
TCOSS version	
TC/LINK-SI license	Key: Expire Date: Registrations:
TCOSS server Name	
Link Type to TCOSS server, transport type PRC or Native	
Secondary TCOSS server Name (for tandem servers only)	
Link Type to secondary TCOSS server (for tandem servers only)	
TC Link User Name	
TC Link User Domain	
TC Link User Password	
Program Id for registration at SAP gateway	
Route String to SAP gateway	
SAP gateway's TCP port or service	
Route String to R/3 application server	
R/3 system number	
R/3 CPI-Cuser : Client (in German "Mandant")	
R/3 CPI-C user : User	
R/3 CPI-C user: Password	