CTI Data Connector

Developer Documentation
for CDC Integration

CDC version: 3.0
Date: May 2010

Table of Contents

1 Introduction

This documentation gives an in depth view, how CTI Data Connector (CDC) is working and how you can implement CDC within your application.

There is an additional documentation, called **Guide for *Step by Step Integration***. This documentation guides you step by step from installation to integration.

A quick and simple approach to CDC would be

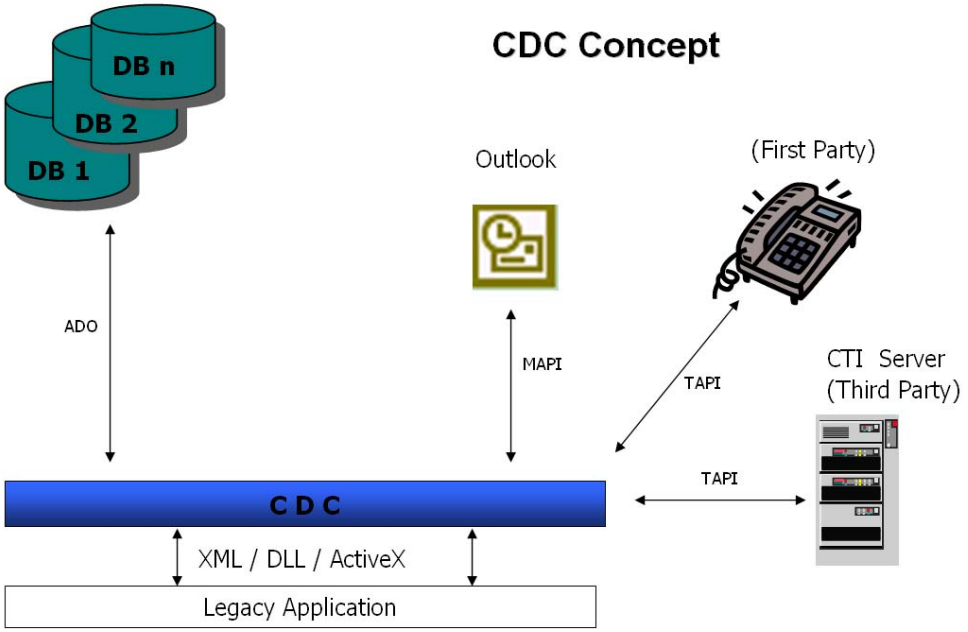
- Read **chapter 1** of this documentation
- Read the *Guide for Step by Step Integration.* The guide makes a lot of references to all important chapters of this documentation

Have a look at our online Knowledge Base. You find additional information regarding this SDK – www.mirage-systems.de/cdc/kb.html - look for CTI Data Connector, English, Integration SDK.

1.1 CDC Concept Overview

CDC acts as a middleware between

- Telephones (First Party solution)
- CTI Servers (Third Party solutions)
- Databases
- MS-Outlook
- Enterprise applications



CDC requires

- A TAPI 2.x driver for communication with telephones or CTI servers
- Databases with ODBC drivers or MS-Access
- MS Outlook or MS-Exchange (optional)

CDC comes with an own front-end (Client) or can be used as a middleware with communication via XML files/DLL/ActiveX calls with legacy applications.

1.2 CDC Technical Overview

CDC consists of two main executables. First, CDC itself (cdc.exe), which is a so called **system tray application** for event handling of phone calls, offered by the computers' TAPI (**T**elephone **A**pplication **P**rogramming **I**nterface). The programme is able to identify the related caller and called ID's and to locate the phone number within configurable databases for displaying the address of the caller or called party. The CDC interface can be suppressed if it is used as a middleware. On any event, signalling a call, CDC saves the related call information, including the address of the caller respective

called party into the XML (E**x**tensible **M**ark up **L**anguage) file **CDCCALLS.XML** for supporting third party solutions. Optionally CDC offers the ability to save a telephone note to this call, which is also stored into a XML file, called **CDCJOBS.XML.** After any XML processing CDC calls are freely configurable executable file with a parameter indicating the new record in those XML files.

Technically, CDC.EXE is an **ActiveX server**. If you plan to make comprehensive CTI integration, then **additionally** use the ActiveX interface – see documentation ***Developer Documentation ActiveX interface*****.**

The second executable is called CDC **Configuration Wizard** (CDCWizz). This programme is used to set up and configure the runtime environment of CDC for each client. This is done by displaying a wizard with an introduction screen and different steps for configuration. The behaviour of the wizard during runtime is fully controlled by the CDC configuration file named CDC.XML and command line parameters.

For editing and displaying the above mentioned XML files during developing we recommend the Microsoft tool XML-Notepad (http://msdn.microsoft.com/xml , search for XML notepad) or download it here http://www.snapfiles.com/get/xmlnotepad.html . If you are using standard editors, bear in mind, that all XML tags are case sensitive. This means, that a XML parser makes a difference between small und capital letters. Only allowed within XML tags are letters and digits. Furthermore, a XML tag must start with a letter. All tags in this documentation are quoted within the smaller (<) and bigger (>) signs, which are not part of the tag name itself.

1.3 Process Description

You can nearly change everything within CDC with just definitions. All definitions are stored in the configuration file CDC.XML, which is located on the server. The user can change some of these definitions with the configuration wizard. The wizard allows definitions per user and writes these definitions in the registry. Some definitions like the programme title or your web address can not be modified by the user with the configuration wizard, but can be modified within the cdc.xml file (only with OEM customization SDK).

CDC is using freely configurable profile definitions for data access. A profile definition is a collection of SQL statements for data access to locate, display and save data to a phone call or a phonebook search. You define with the SQL statements which tables and fields are holding the phone numbers for identification and what is/are the corresponding primary key(s). This key is used to locate the related address and/or the contact person.

On any call event, CDC tries to locate the phone number to get the primary key to the address/contact person. On success it pops up with the caller party information (showing the address). If an exact search will return no caller party, CDC tries to locate the caller party by cutting off up to four digits on the end of the phone number (configurable). In this case or if the exact search delivers more than one

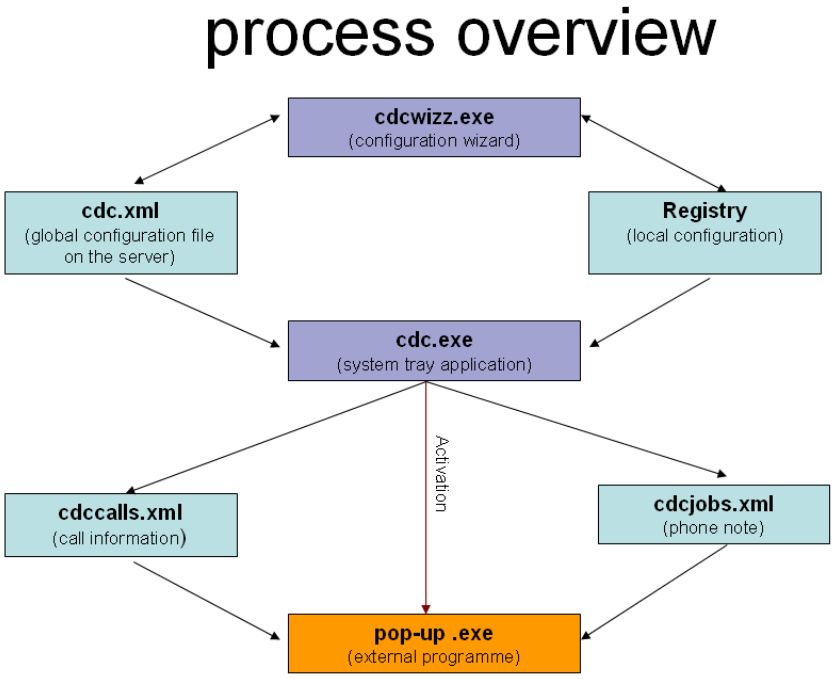
result, a choice of maximum five possible caller parties per profile will be displayed and the user can make his own selection.

Additionally to the location of the caller party, CDC

- writes information about the call into the file **cdccalls.xml**
- calls an external programme (your software)

with the unique TAPI handle of this call as a command line parameter. Your software can read this information in cdccalls.xml and pop up with the address of the caller.

A further function of CDC is the saving of a telephone note to the actual call if it was activated. On the event of the saving (the user is pressing the SAVE or CLOSE button) a new record with this information in the telephone note file CDCJOBS.XML is created. Again the external programme with the unique key of this job is called.



1.4 Handling Phone Numbers

The phone numbers within the database fields may be entered in any format. CDC takes care of phone numbers and stores it in an international format (e.g. 004970054885342) in a separate table.

CDC uses **two internal tables for each database**.

- **CDCCTI** - table with phone numbers and index keys to the address and contact person table
- **CDCCTIHELP** – temporary table – only used during data preparation

An interval can be specified, when all phone numbers of a database are copied to the CDCCTI table.

1.5 Using Microsoft Outlook

CDC can use addresses from the MS-Outlook contact folder(s) (local or from MS-Exchange). The phone numbers are stored in the same way like data from other databases in the CDCCTI table – see section *process description.* The only significant difference is that the tables CDCCTI and CDCCTIHELP are created during installation in a local access database. You find the access database in the folder **…\documents and settings\username\application data\application exe name\cdc.mdb.** The synchronisation process is done in the background and no user interaction is required.

1.6 Folders used with CDC

A normal installation of CDC is as follows:

- All files are on the server in a folder **…\cdc**
- Files generated at runtime with local data (e.g. data of the actual call) are in a local folder **…\documents and settings\username\application data\application exe name**

2 Necessary Steps for Integration

There are only a few steps necessary for a basic integration:

- Implement Outgoing calls (via DLL, call EXE, via file et.c)
- Implement Incoming calls – Step 1 - define SQL statements
  Step 2 – integration in an application
- Implement storing a phone note or activity when using the CDC Client interface

If you are a software vendor, then you can customize the application with the OEM SDK – *Developer Documentation for OEM Integration*

If you do have a Browser based software see chapter *Implementation for Browser based software*

3 Implement Outgoing Calls

This is quite easy. There are several ways to do the implementation:

- Dial with Hotkey. The user can dial with highlighting a phone number and pressing a Hotkey - no programming is necessary
- Dial via Hyperlink
- Dial with right mouse click within the Browser (only available within Internet Explorer) – no programming is necessary
- Use the right mouse button and display an option *dial.* If the user selects dial, you have to read the field content (= phone number) and you call CDC with the field content. An other possibility is to show all phone numbers of the address instead of the field content
- Make a dialler button like  next to each field containing a phone number
- Make a dialler option in the menu bar of the address windows or use function keys. When the dialling option is selected, all phone numbers are displayed (including the phone numbers of the contact persons)
- If you already have implemented outgoing calls via the Windows Dialler, you don't have to make any changes. Just configure CDC that all outgoing dialling events going to the Windows Dialler go to CDC (default setting). You have to set *Registerrequest=1* in the setup node of the cdc.xml file. See chapter *Configuration file cdc.xml*

We recommend using the right mouse click method, because it is easy and can be used generic.

There are different methods to implement dialling with CDC, depending on your development environment and operating system.

3.1 Use a DLL call

The simplest method is to use a DLL call.

Integrate **cdctapi.dll** and **cdcconfig.dll** in your project. Then you can use the following code to dial:

```
Dim oClient As Object
Dim oConfig As Object
Dim bOK As Boolean
Dim sTel As String
Dim lHwnd As Long

  If bOK Then
     ' --- sTel is the phone number you want to dial – e.g. 0044-6047/6250
    '—format the phone number for correct dialling
```

```
    sTel = oClient.TranslateAddress(sTel, 3)
        ' --- dial via CDC
    lHwnd = oClient.lineMakeCall(sTel)
  End If

Private Sub Form_Load()
  Text1.Text = ""
  Set oConfig = CreateObject("CDCConfigLib.CDCConfig")
  Set oClient = CreateObject("CDCTAPILib.CDCClient")
  bOK = oClient.bInitialize(oConfig)
End Sub
```

You find a demo application *dial.exe* in the SDK with the Source Code in VB. The application is in the folder *..\demoapplication\Demo Outgoing Calls.*

3.2 Use a Hyperlink to dial

You can dial with a Hyperlink in Browser applications. To use this functionality, the application has to be modified. Dialing via Hyperlink allows dialing just with a click.

To dial via a hyperlink in a Browser based application, just insert the following command in your application:

phonenumber

where phonenumber has to be replaced by the real phone number like

+4970054885342

It is displayed on the screen as follows: +4970054885342

To enable dialing via hyperlink open the Configuration Wizard and check *Activate dialing via Hyperlink*.

This function is available in all Browsers (e.g. Internet Explorer or Firefox).

3.3 Use DIAL.EXE

If your application **doesn't support DLL functions** or you don't' like to integrate dll's in your application, you can use dial.exe for outgoing calls. You find the application DIAL.EXE in the SDK folder (with Source Code). Start dial.exe with a phone number as parameter

dial.exe 0049/700548853-42

Dial.exe handles the call to CDC and ends, without showing a front end. The application is in the folder *..\demoapplication\Demo Outgoing Calls.*

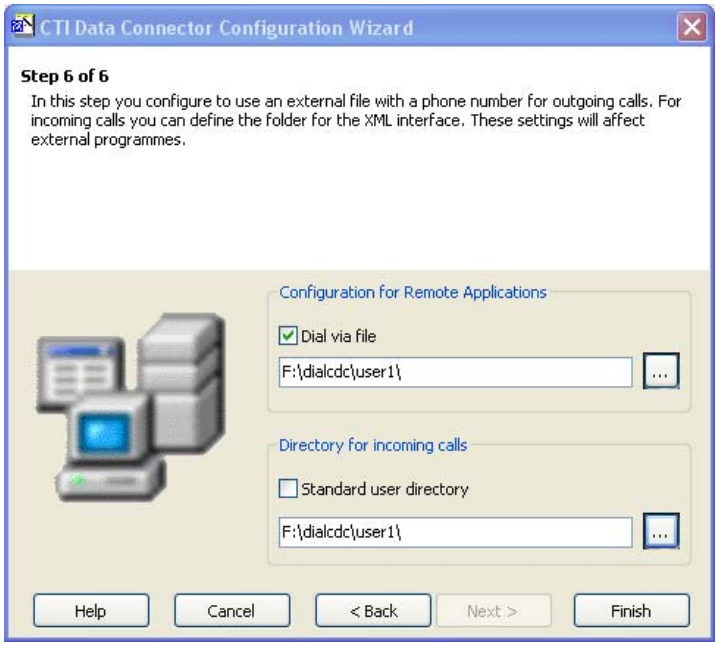
3.4 Use a File for Dialling

If your application is running on a Server (e.g. Terminal Server) or I-Series, you could write the phone number in a XML File. CDC polls for this file, reads the number, deletes the file and dials the number.

The user has to select the directory in the configuration wizard. For every user this has to be a separate folder, because the filename is always **dial.xml.**

The format of the file is:

```
<CDC>
   <DialPhoneNumber>PHONENUMBER</DialPhoneNumber>
</CDC>
```



The result of the caller identification process could also be written to the same directory. This can be configured in the wizard.

If your application runs on a terminal server we strongly recommend that you implement the file method as an alternative method. CDC could then run on the local machine and the First Party solution (phone directly connected to the PC) is supported even in the Terminal Server mode.

3.5 Advanced Dialling Control

If you want to control the call process (e.g. hold or terminate a call) then use the ActiveX interface - see documentation *Developer Documentation ActiveX interface.*

3.6 How does CDC handle the dialling?

CDC does everything necessary:

- CDC formats the number for outgoing dialling, e.g. #004460476250 using the configuration settings done via the configuration wizard and in US/Canada using dialling rules
- CDC makes the necessary TAPI calls
- CDC locates the phone number in it's table to get the unique address key
- CDC opens the address table with this key to get further information to this address
- The unique address key of the located call party with further information is written in cdccalls.xml
- If you specified an external programme in the main node <profile>, then this external programme is executed to inform your application that a call has occurred

A caller identification is done with every outgoing call. Depending on the configuration setting in the wizard, the outgoing call is signalled or not. We recommend signalling the call, because this is even done if the user dials with his phone set. You could then pop-up the customer folder.

The Exe file, which you defined (e.g.PopUp.exe) is activated again. This is to give you the possibility to make actions for outgoing calls. If you just want to dial outbound in your application then ignore *<JobMode> = 2* in application.

Note: If working with the simulator and using dial.exe or a DLL call, the Simulator just shows the number you have dialled, but doesn't give the event to CDC. If you make an outgoing call with the CDC Client, then CDC gets the event and does everything like described above.

4 Implement Incoming Calls

CDC can use any SQL Database or MS-Access to perform online caller identification.

If only the caller data should be displayed within the Desktop Notification window, then step 1 is sufficient. If the caller data should pop-up in an application, then step 2 is required.

4.1 Step 1 – Add tables and define SQL Statements

To implement incoming calls, you just have to define some SQL Statements and add 2 tables in your database.

4.1.1 Add tables

First, add in your database the 2 tables described in the chapter: *Handling phone numbers*

```
CREATE TABLE CDCCTI (IDCti VARCHAR(50) PRIMARY KEY,
                     IDXAdr VARCHAR(254) NULL,
                     IDXAsp VARCHAR(254) NULL,
                     Nummer VARCHAR(50) NULL);

CREATE TABLE CDCCTIHELP (IDCti VARCHAR(50) PRIMARY KEY,
                     IDXAdr VARCHAR(254) NULL,
                     IDXAsp VARCHAR(254) NULL,
                     Nummer VARCHAR(50) NULL);

GRANT SELECT, INSERT, UPDATE, DELETE on CDCCTI to Public;
GRANT SELECT, INSERT, UPDATE, DELETE on CDCCTIHELP to Public;
```

You can change the name of these tables. In the file cdc.xml add an entry for **each profile**.

<Profile>.....
< CDCCTITable>My_Table _1 </CDCCTITable>
<CDCCTIHELPTable>My_Table_2 </CDCCTIHELPTable>

To store the CTI tables in **another database**, you have to create a View in the main database (sample for MS-SQL)

```
CREATE VIEW [dbo].[CDCCti]
AS
SELECT  IDCti, IDXAdr, IDXAsp, Nummer
FROM   CDCDB.dbo.CDCCti
```

```
CREATE VIEW [dbo].[CDCCtiHelp]
AS
SELECT  IDCti, IDXAdr, IDXAsp, Nummer
FROM   CDCDB.dbo.CDCCtiHelp
```

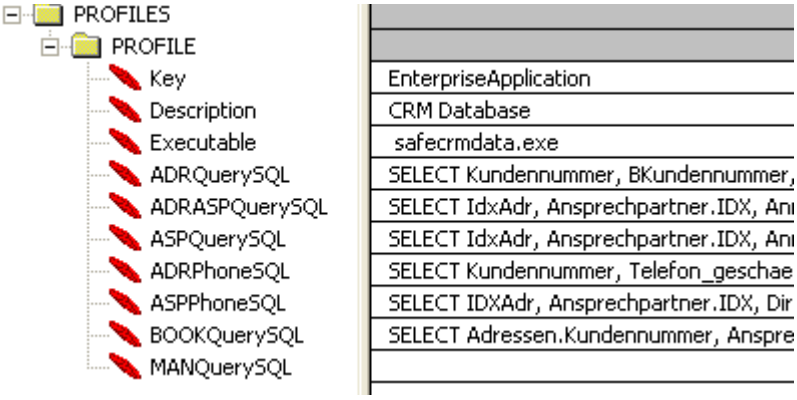
If you changed the tables names using the configuration option < CDCCTITable>, the View must have exactly the same name like defined with < CDCCTITable>.

4.1.2 Define SQL statements

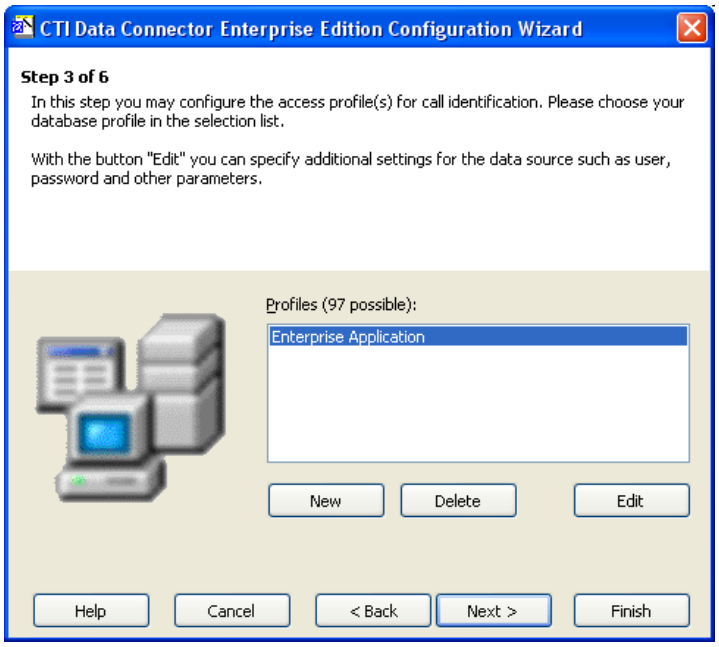
The necessary SQL statements are defined within the file **CDC.XML in the node <PROFILES>.** We recommend making a backup of this file before editing it.

This section holds 1 to n profile definitions within XML elements under their separate sub nodes. The sub nodes' tag name must be *<PROFILE>*.

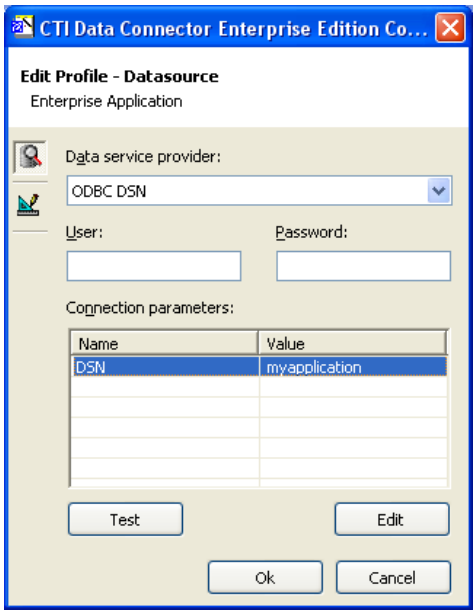
With profiles you define the access to a database. For each database a profile is needed.



The SQL statements can either be modified with an XML Editor or with the configuration wizard of CDC. Go to the page, where the database profiles are displayed.

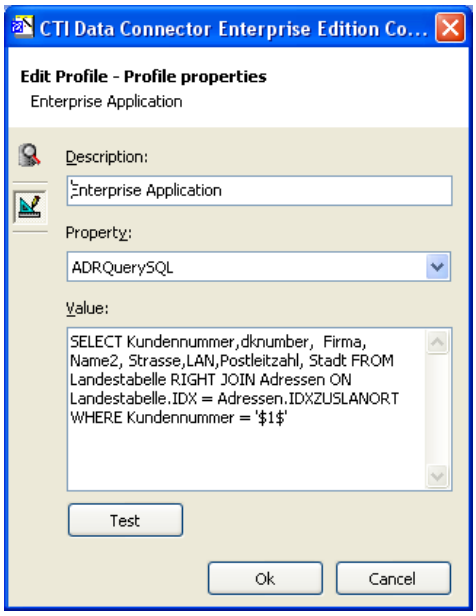


Choose Edit



Select your connection parameters and press test. The connection to the database is tested.

Press the button  to display the first SQL statement.



You can modify the SQL statement and test it online on your database. Change all SQL statements listed in the field Property and test every SQL statement.

The name of the database profile can only be changed in the XML file.

Below, you find a detailed explanation of every SQL statement. The fields **highlighted in green** can be edited in the configuration wizard

Elements of a profile definition in cdc.xml

Element name	Description
<Key>	Unique key for this profile definition, e.g. the name of the legacy application
<AddPhoneNote>	Allows to type in a phone note – see chapter *5 Using the CDC Client to store a Phone Note and Activity* 0 = No – don't show this option 1 = Yes – allow to type in a phone note
<Executable>	The name of a windows executable programme, optionally with the path to the file. If there is no path specified, the programme must be located within the CDC directory. This executable is called by CDC after any call event with a parameter indicating a call or telephone note save event, **providing an address of the database, defined in the <key> node.** This is the programme, where you have to handle all calls if CDC is running as a middleware and is referenced in this documentation as *external programme* If you don't specify an executable, the executable defined in the node <defaults>, Executable is processed.
<Executable>	The name of a windows executable programme, optionally with the path to the

file. If there is no path specified, the programme must be located within the CDC directory. This executable is called by CDC after any call event with a parameter indicating a call or telephone note save event, **providing an address of the database, defined in the <key> node.** This is the programme, where you have to handle all calls if CDC is running as a middleware and is referenced in this documentation as *external programme*

If you don't specify an executable, the executable defined in the node <defaults>, Executable is processed.

To enable caller identification, the following SQL statements are necessary:

- <ADRQuerySQL>
- <ADRASPQuerySQL>
- <ASPQuerySQL>
- <ADRPhoneSQL>
- <ASPPhoneSQL>

<ADRQuerySQL>

The SQL statement for displaying and identification the address of the other call party in CDC. It is used to have a link from the internal table *cticdc* to the address table.

The first field must return the unique address key in the address table. The second field may return the customer ID. Fields 3 and 4 are interpreted as company name 1 and 2, field 5 as street and fields 6, 7 and 8 as country, postal code and city. Fields 3 to 8 are used for displaying an address. You can swap fields (e.g. field 7 and field 8) or combine fields e.g. field 8 = city + ' ' + state. If you want to omit a field use NULL instead.

'1' is a placeholder for the unique address key in the address table.

Example: SELECT IDXAdr, CustomerID, CompanyName, Null, Street, Country, ZipCode, City FROM Address WHERE IDXAdr = '1'.

This indicates **IDX Adr AS** unique address key, CustomerID AS CustomerID and so on. All following queries are built in the same way.

<ADRASPQuerySQL>

The SQL statement for displaying and identification of **all contact persons** to the address. The first field should return the foreign key of the address table, the second field must return the unique contact key in contacts table. Fields 3, 4, 5 and 6 are interpreted in the following order salutation, title, first name, last name.

'1' is a placeholder for the link from contacts to the unique address key in the address table.

Example: SELECT contacts.ADRIndex, contacts.IDXContacts, contacts.Salutation, contacts.Title,

contacts.FirstName, contacts.LastName Where contacts.ADRIndex ='1' ORDER BY contacts.LastName

<ASPQuerySQL> The SQL statement for displaying and identification of **one contact person** to the address. It is used to have a link from the internal table cticdc to the contact person.

The fields are interpreted in the same order like above.

'1' is a placeholder for the link from contacts to the unique address key in the address table. '2' is a placeholder for the unique key in the in the contact person table.

Example: SELECT contacts.ADRIndex, contacts.IDXContacts, contacts.Salutation, contacts.Title, contacts.FirstName, contacts.LastName Where contacts.ADRIndex ='1' and contacts.IDXContacts ='2' ORDER BY contacts.LastName

<ADRPhoneSQL> The SQL statement for preparing the phone numbers of an address. The first field must return the unique address key in the address table, all other fields should contain phone numbers for preparing to identify them in case of call events.

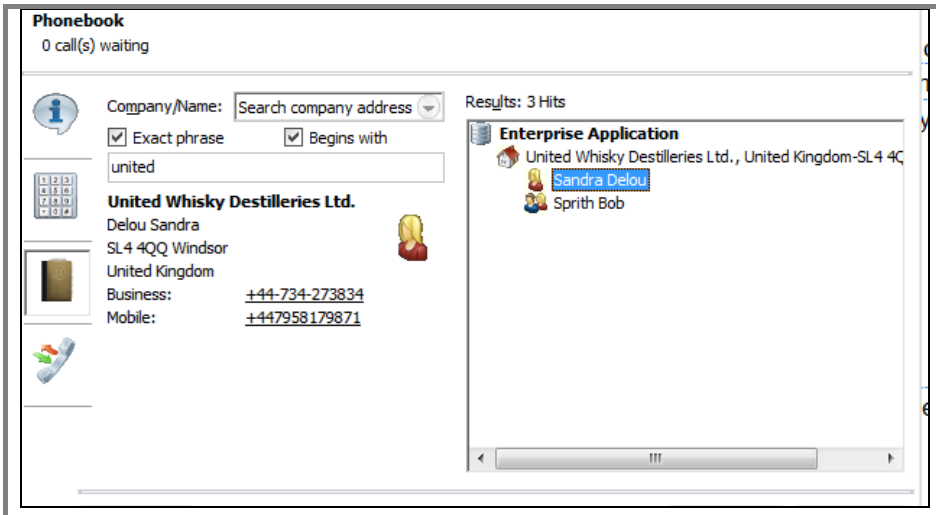
Example: SELECT IDXAdr, phone1, phone2, fax, mobile, pager from address

<ASPPhoneSQL> The SQL statement for preparing the phone numbers of a contact person. The first field should return the foreign key of the address table, the second field must return the unique contact key in contacts table. All other fields should contain phone numbers for preparing to identify them in case of call events.

Example: SELECT ADRIndex, IDXContacts, directnumber, assistant, fax, mobile, pager, home from Contacts

To enable the s**earch option in the address book**, the following SQL statements are necessary:

- <ADRASPBOOKQuerySQL>
- <ADRBOOKQuerySQL>
- <ASPBOOKQuerySQL>
- <BusinessCardAspSQL>
- <BusinessCardAdrSQL>



<ADRASPBOOKQuerySQL> The SQL statement for searching an **address** or contact name within the phonebook.

The first field must return the unique address key of the address table. The second field must return the unique contact key in contacts table.

Fields 3, 4, 5, 6 are interpreted as company name 1, country, postal code and city.

Fields 7, 8 and 9 are interpreted as contact last name, first name and gender.

1 will be replaced by the search term.

Example: SELECT address.IDXAdr, contacts.IDXContacts, address.CompanyName, address.Country, address.ZipCode, address.Town, contacts.LastName, contacts.FirstName + ‘ - ‘ + coalesce(contacts.Salutation,’ ’) + ’ ‘ + coalesce(contacts.Title,’ ’), contacts.Gender FROM address, contacts WHERE address.IDXAdr = contacts.IDXAdr and address.CompanyName LIKE ‘1’ and contacts.LastName LIKE ‘1’ ORDER BY address.CompanyName, contacts.LastName, contacts.FirstName

<ADRBOOKQuerySQL> The SQL statement for searching an **address** or contact name within the phonebook. It is nearly identical to <ADRASPBOOKQuerySQL>. The only difference is in the WHERE condition.

Example: SELECT address.IDXAdr, contacts.IDXContacts, address.CompanyName, address.Country, address.ZipCode, address.Town, contacts.LastName, contacts.FirstName + ‘ - ‘ + coalesce(contacts.Salutation,’ ’) + ’ ‘ + coalesce(contacts.Title,’ ’), contacts.Gender FROM address, contacts WHERE address.IDXAdr = contacts.IDXAdr and address.CompanyName LIKE ‘1’ ORDER BY address.CompanyName, contacts.LastName, contacts.FirstName

<ASPBOOKQuerySQL> The SQL statement for searching a **contact name** within the phonebook. It is nearly identical to <ADRASPBOOKQuerySQL>. The only difference is in the WHERE condition.

Example: SELECT address.IDXAdr, contacts.IDXContacts, address.CompanyName, address.Country, address.ZipCode, address.Town, contacts.LastName, contacts.FirstName + ' - ' + coalesce(contacts.Salutation,' ') + ' ' + coalesce(contacts.Title,' '), contacts.Gender
FROM address, contacts WHERE address.IDXAdr = contacts.IDXAdr and contacts.LastName LIKE '1'
ORDER BY address.CompanyName, contacts.LastName, contacts.FirstName

<BusinessCardAspSQL> The SQL statement for displaying the **address and contact person details** of the selected item in the **search result**

The first four fields must return the data for the **first four lines in the business card.**

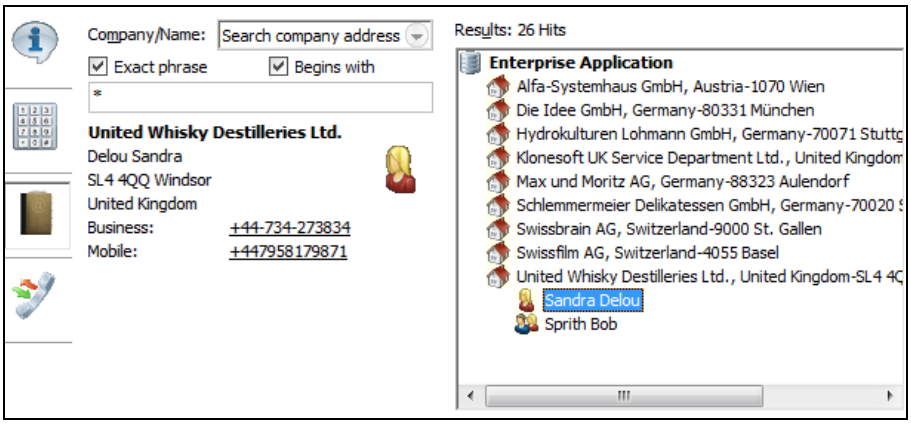
These four fields can be the name of a company, contact person name, postal code with the city and the country.

The following fields are reserved for the phone numbers to display in the business card.

There is a maximum of twelve phone number fields although the user interface can display only 8 at a time. If a phone field is empty the next one of the SQL statement is displayed.

The labels for these fields has to be defined with <NumbertextAspXY> in the cdc.xml file

'1' is a placeholder for the unique address key in the address table.
'2' is a placeholder for the unique contact key in the contact table.



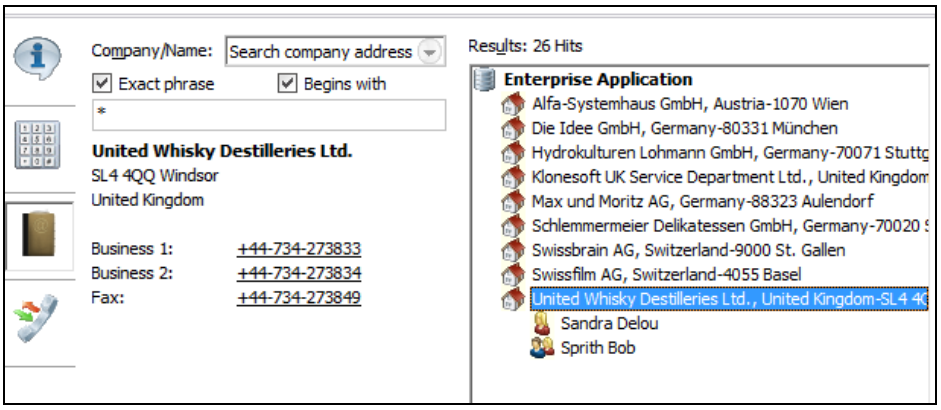
```
Example: SELECT      TOP 1
                company,
                contact_first_name + ' ' + contact_last_name,
                postal_code + ' '+ city,
                country,
                phone_business1,
                phone_business2,
                phone_private,
                mobile,
                fax,
                contact_phone,
                contact_mobile,
                contact_personal_phone,
                contact_personal_mobile
        FROM    account,
                contacts
        WHERE           IDXaccount = index_account and
                index_account = '$1$' and
                index_contacts = '$2$'
```

<BusinessCardAdrSQL>

The SQL statement for displaying the **address** information of the selected item in the **search result**

The definition is similar to **<BusinessCardAspSQL> .**

The first four fields can be the name of a company, address2, postal with the city and the country.



Example:

```
SELECT      TOP 1
            company,
            address2,
```

```
            postal_code + ' ' + city,
            country,
            phone_business1,
            phone_business2,
            phone_private,
            mobile,
            fax
        FROM  account
        WHERE index_account = '$1$'
```

<MANQuerySQL> The SQL statement for returning clients (of the own company), if there any. The first field is interpreted as the unique key, the second field is interpreted as the client name. Normally, this statement is omitted.

Example: SELECT ClientIndex, ClientName FROM Clients

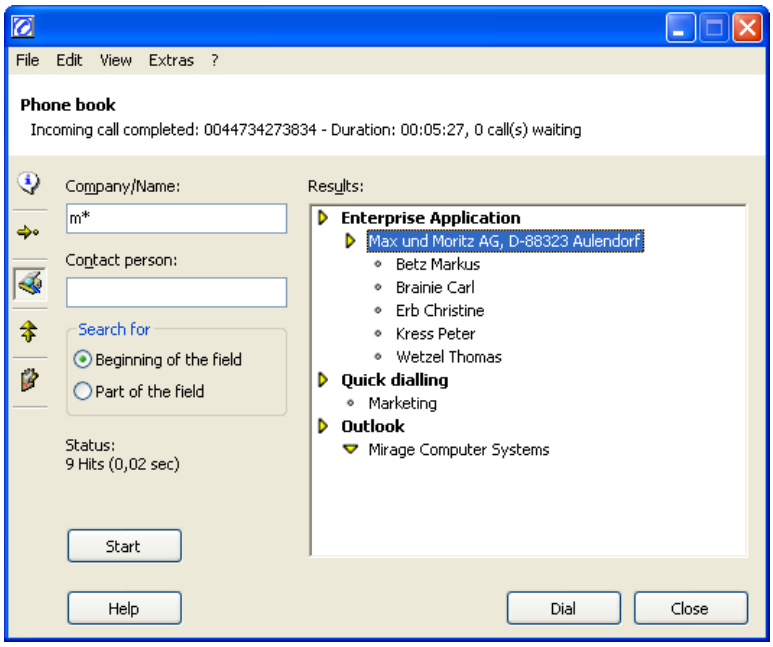
The statements in the elements <ADRQuerySQL>, <ADRASPQuerySQL>, <ASPQuerySQL> and <BOOKQuerySQL> must contain a WHERE section to locate the required record by primary key. Within this WHERE section the parameter 1 is allowed for the unique address key, respectively address name in case of the <BOOKQuerySQL>. The parameter 2 is allowed for the unique contact key respectively contact name in case of the <BOOKQuerySQL>. Both parameters are replaced by the programme with the required search term.

All elements may be stated, but it is not a must. Also, all fields within the SQL statement may be stated, if an interpreted field is not supported by the table it could be passed by stating NULL for the field.

Have a look at our online Knowledge Base. You find SQL samples if you do not have a contact person table or 1:n phone numbers per table – www.mirage-systems.de/cdc/kb.html - look for CTI Data Connector, English, Integration SDK.

4.1.3 Test the SQL Statements

You can test, whether the statements are logically correct by starting CDC, selecting the phone book and type in the search string *. All addresses should be displayed. Click on an address. You should see the contact person. Click on the contact person, you should see the phone numbers.


File Edit View Extras ?
Phone book
Incoming call completed: 0044734273834 - Duration: 00:05:27, 0 call(s) waiting
Company/Name:
m*
Contact person:
Search for
Beginning of the field
Part of the field
Status:
9 Hits (0,02 sec)
Start
Help
Results:
Enterprise Application
Max und Moritz AG, D-88323 Aulendorf
Betz Markus
Brainie Carl
Erb Christine
Kress Peter
Wetzel Thomas
Quick dialling
Marketing
Outlook
Mirage Computer Systems
Dial
Close

4.2 Step 2 – Integration in your Application

By defining the SQL statements, the complete caller identification is already working as follows:

- CDC identifies the caller
- Displays caller data in Desktop Notification window
- Writes the caller data in the file **cdccalls.xml**
- Invokes a predefined EXE file

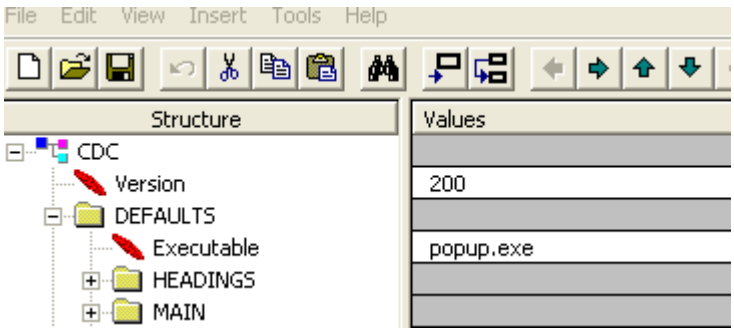
4.2.1 Invoke/Inform your application about a call

Each time a call is processed, CDC calls an executable file (your application) if you use CDC as a middleware.

You can specify a separate programme for each database or one global application for all databases. This is done in the cdc.xml file.

Global programme handling

If you want to specify 1 programme, that handles **all events for all databases including MS-Outlook**, this has to be specified in the <Defaults> section with **Executable**

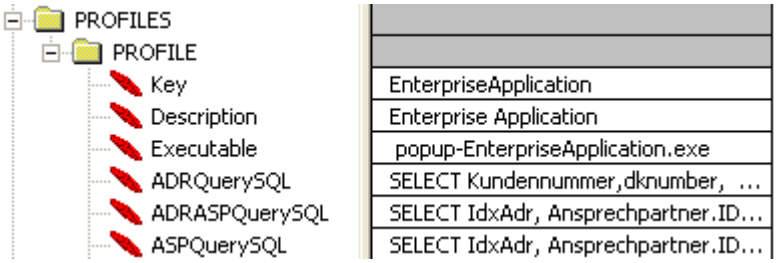


The database, where the address is stored, can be retrieved from the cdccalls.xml file from the entry *<Profile>.* This enables you to write a global programme which acts different, depending on the source of the address.

If a specific Executable is defined for a database or MS-Outlook, then that Executable is used.

Specific programme for every database

If the events should be handled depending on the database, where the address is stored, then this can be specified in the <profiles> node with the entry **Executable** for each profile/database.



Even for MS-Outlook, an individual POP-Up screen could be designed (if the inbuilt feature to open a MS-contact folder is not used). The Exe file is specified in the <Outlook> node.

If you use one database and MS-Outlook, it is sufficient to use one external programme defined in the defaults section.

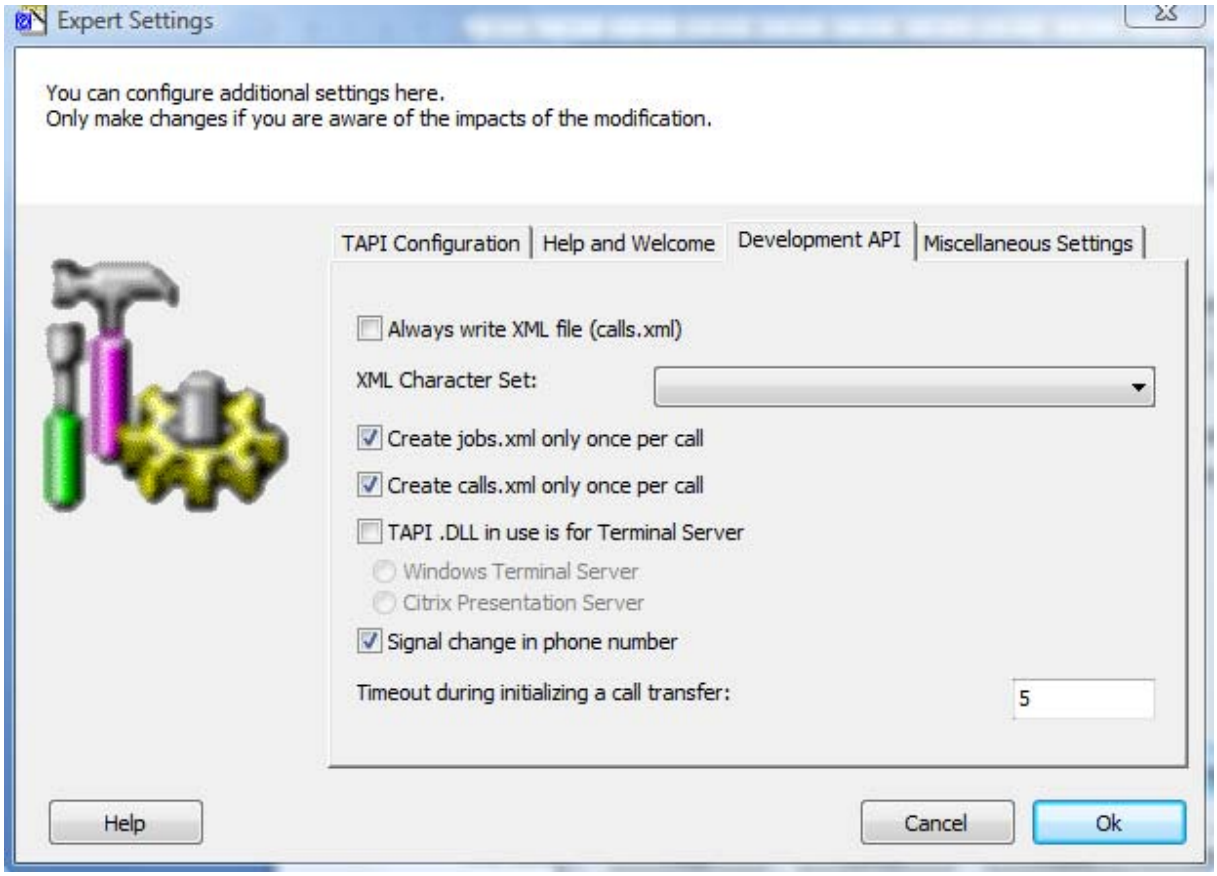
4.2.2 Event Handling

There are 4 events, when the file cdccalls.xml is created and the external programme is started. These events are passed in the entry **<JobState>,**

- Ring
- Call Active
- Call Terminated
- Call on Hold

It can be defined, whether the cdccalls.xml is created **each time an event occurs** or just **only once**. This is defined in the cdc.xml file, node <main>, **SaveCallOnlyOnce**. Only once is the default handling unless your application explicitly wants to be informed about each event. Informing about each event is only necessary if you want to save information about a call like start and end time, duration or if you want to display a status like call on hold on your application form.

The user can change the event handling in the configuration wizard, page *Expert Settings.*



When the Desktop Notification window is used, then the cdccalls.xml file is only **created when the user clicks on the Desktop Notification window** to get the details of the caller (default).

If you **always need** the caller information (e.g. to protocol every call), then configure *Always write XML file* in the configuration wizard. You can then use the item <*FadeInWasClicked*> in the cdccalls.xml file to find out, if the user has clicked on the Desktop Notification and wants to view the data in your application.

CDCCALLS.XML holds for any call appearance one record under the root node <*CDC*>. **It is in the duty of the called programme to delete the record** if it is of no further use. If multiple calls occur at the same time or the file is not deleted, all calls are added within the file.

The record is indicated by the unique TAPI handle as main node under which all related call information are collected.

CDC calls the external programme with the command line parameter ***-cCALL_n***, where n is the long value of the unique TAPI handle for this call – entry <*hTapiCall*>. The parameter followed by the ***–c*** is also the main node name of the record. Be aware, **that the executable file may be called for any event** (see above).

If the user works with the CDC POP-UP Window for incoming calls (not recommended when used as a middleware) then even for the same event the exe could be called several times. The first time, the event occurs with the information of all possible caller parties and the second time - if the user selects one caller party - with the selected caller party.

You therefore have to check, if the event is from the same caller (= the same TAPI handle *<hTapiCall>*) and display the caller only once. The additional information could be used to store the duration of the call or to signal on the interface, that the call is active or was terminated.

If the Desktop Notification Window is used, the first event could be a Call Active or Call Terminated, depending when the user clicks on the Desktop Notification window.

You find a sample application with source code in the folder *Demo Incoming Calls* of the SDK.

4.2.3 The file CDCCALLS.XML

The file is in the folder **…\documents and settings\username\application data\ application exe name. (**this is the default path which can be configured within the wizard or in the node <setup> value *<XML Directory>)*.

You can access this local folder via Windows API:

```
Private Declare Function SHGetPathFromIDList Lib "shell32" Alias "SHGetPathFromIDListA" (ByVal pidl As Long, ByVal pszPath As String) As Long

Private Declare Function SHGetSpecialFolderLocation Lib "shell32" (ByVal hwndOwner As Long, ByVal nFolder As Long, pidl As Long) As Long

Private Declare Sub CoTaskMemFree Lib "ole32" (ByVal pv As Long)

Private Const CSIDL_APPDATA = &H1A

Public Function gsGetAppDataFolder(Optional ByVal sSubFolder As String = "") As String

   Dim sPath As String
   Dim lpidl As Long

   gsGetAppDataFolder = ""
   If SHGetSpecialFolderLocation(0, CSIDL_APPDATA, lpidl) = 0 Then
      sPath = Space$(260)
      If SHGetPathFromIDList(ByVal lpidl, ByVal sPath) Then
         gsGetAppDataFolder = Left(sPath, InStr(sPath, Chr$(0)) - 1)
         If sSubFolder <> "" Then
            gsGetAppDataFolder = gsFitPath(gsGetAppDataFolder) & sSubFolder
         End If
      End If
      CoTaskMemFree lpidl
   End If

End Function

Public Function gsFitPath(sPath As String) As String
   If Right(sPath, 1) <> "\" Then
      gsFitPath = sPath & "\"
   Else
```

```
    gsFitPath = sPath
  End If
End Function
```

This file is only generated at run time if you have specified an external programme (see above).

Elements of cdccalls.xml

Element name	Description
<JobKey>	A unique key indicating the job for saving
<hTapiCall>	The unique TAPI handle of the call
<JobMode>	1 indicates an inbound call, 2 indicates an outbound call
<JobState>	Signals the event 1 = Ring 2 = Call Active 3 = Call Terminated 4 = Call on Hold
<ForeignTel>	The phone number of the other call party. This field is left blank if no phone number has been transmitted from the switch. The phone number is signalled exactly in the way the TAPI driver delivers it – e.g. 070054885342 We have simplified the information of the TAPI driver which is signalled in the value CallerID and CalledID. <ForeignTel> always signals the number of the other party/subscriber. <u>Incoming call</u> CallerID = ForeignTel CalledID = OwnerTel <u>Outgoing call</u> CallerID = OwnerTel CalledID = ForeignTel
<CallerIDInternational>	Phone number in international format – e.g. 004970054885342
<OwnerTel>	Phone number which was called
<Profile>	The profile key of the profile definition where the caller party was located. This field is left blank if more than one possible call party are located.
<ADRKey>	The unique address key of the located call party. This field is left blank if more than one possible call party are located.
<ADRText>	The address in clear text, rows are separated by a CR/LF. This field is left blank if more than one possible call party are located.
<CustomerID>	The customer ID of the address. This field is left blank if more than one possible call party are located.
<CustomField>	Individual value which can be set using the ActiveX interface. Can be

	used to identify an record – e.g. customer number or own ID
<ASPKey>	The unique contact person key of the located call party. This field is left blank if more than one possible call party are located.
<ASPText>	The contact person in clear text. This field is left blank if more than one possible call party are located.
<ASPFound>	Indicates with –1 that minimum one contact person has been found
<ADRFound>	Indicated with –1 that the address has been found
<Date>	Date of the call
<Time>	Time of the call
<StartTime>	Time, when a call has started
<EndTime>	Time, when a call has ended
<Activated>	indicates, if a call was successful (connection was established) - 1 = Yes, 0 = No
<Completed>	indicates, if a call was completed - 1 = Yes, 0 = No
<IsNumberChangedEvent>	indicates, if the phone number has changed during the active call. This happens if a call is transferred
<FadeInWasClicked>	indicates, if the user clicked on the desktop notification
RedirectingID	Phone number which redirected the call (The redirecting party identifies the address which redirect the session)
RedirectionID	Phone number to whom the call was redirected (The redirection party identifies the address to which the session was redirected)
<Choices>	Sub records of the choices displayed to the user if more than one possible call party are located. These records are not delivered if only one call party is located, or the user already selected one call party
<Contacts>	Sub records of contact persons if there are more than one found.

Example file

	<JobKey>**20020123155640253814**</JobKey>
	<hTapiCall>**16047**</hTapiCall>
Incoming call	<JobMode>**1**</JobMode>
Call is active	<JobState>**2**</JobState>
Phone number	<ForeignTel>**0044734273833**</ForeignTel>
Database	<Profile>**Application 1**</Profile>
Database index of the address	<ADRKey>**11995106183752**</ADRKey>
Address in plaintext	<ADRText>**United Whisky Destilleries Ltd. Freemont Tower 3 Perkens Bridge GB-Windsor SL4 4QQ**</ADRText>
Customer number	<CustomerID>**D200004**</CustomerID>
Database index of the contact person	<ASPKey>**102000032012145314**</ASPKey>
Contact person in plaintext	<ASPText>Mr. Bob Sprith</ASPText>
Additional information of the call	<ASPFound>**-1**</ASPFound>
	<ADRFound>**-1**</ADRFound>
	<Date>**23.01.2002**</Date>
	<Time>**15:42:56**</Time>
	<Choices />
	<Contacts />

Elements of the sub record choices

Element name	Description
<ProfileKey>	The *unique profile key* indicating the profile definition where the possible call party has been found.
<ProfileText>	The *profile name* displayed to the user, where the possible call party has been found.
<Addresses>	Up to five *address sub records* where the possible address are stored

Elements of the sub record addresses

Element name	Description
<ADRKey>	The unique address key of the located call party.
<ADRText>	The address in clear text, rows are separated by a CR/LF.
<Contacts>	Sub records of *all contact persons* found to this address

Elements of the sub record contacts

Element name	Description
<ASPKey>	The unique contact person key of the located call party.
<ASPText>	The contact person in clear text.

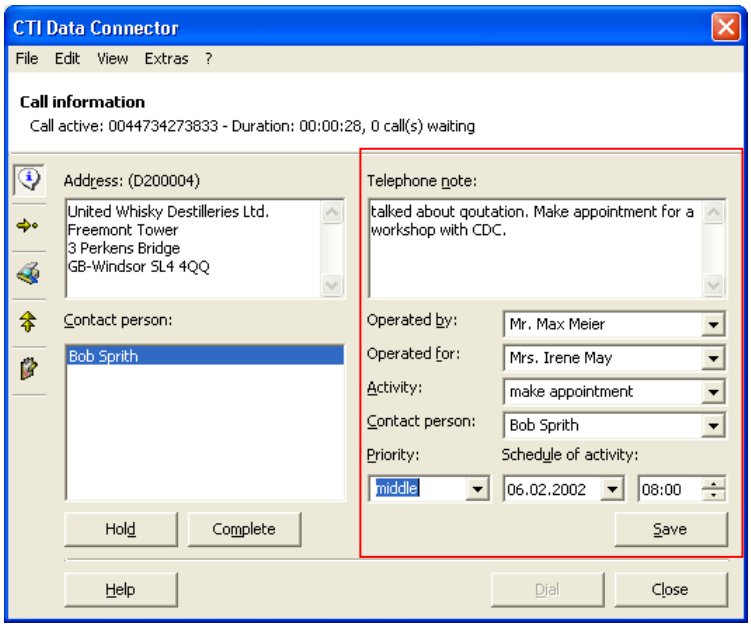
5 Using the CDC Client to store a Phone Note and Activity

If CDC is used in a stand alone mode, phone notes and information for further activities can be save in the legacy application.

The CDC Client could be used when:

- Your application is not active
- For users, which have no access to your application

You need to have a valid license (CDC Extended Edition) to use the client interface. The client interface can be activated/deactived from your application with the **registry ITEM IconWithOutClient**. Values see chapter *sub note <setup>.*

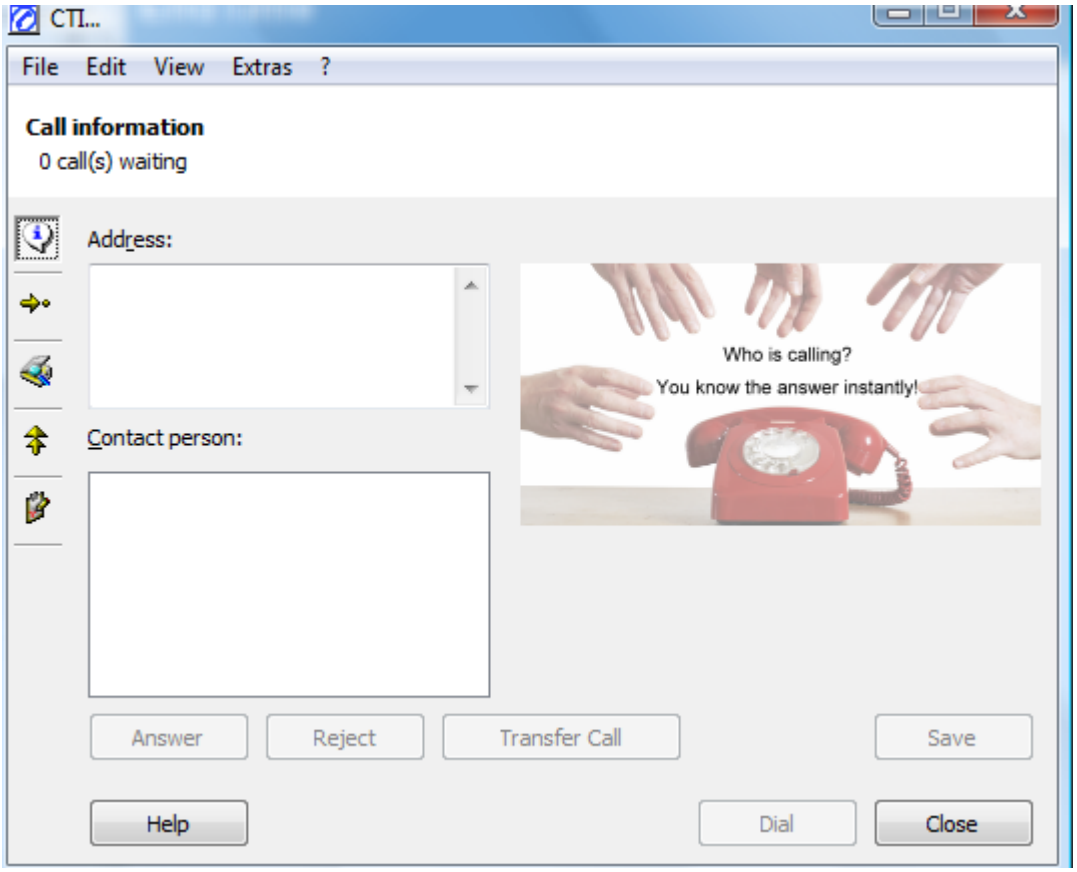


When the user presses the SAVE or CLOSE Button, the following actions are performed

- the information is written in the cdcjobs.xml file
- the external programme, specified in the main node <profile> in cdc.xml, is called
- The programme reads the information in cdcjobs.xml
- The programme writes the information into the legacy database

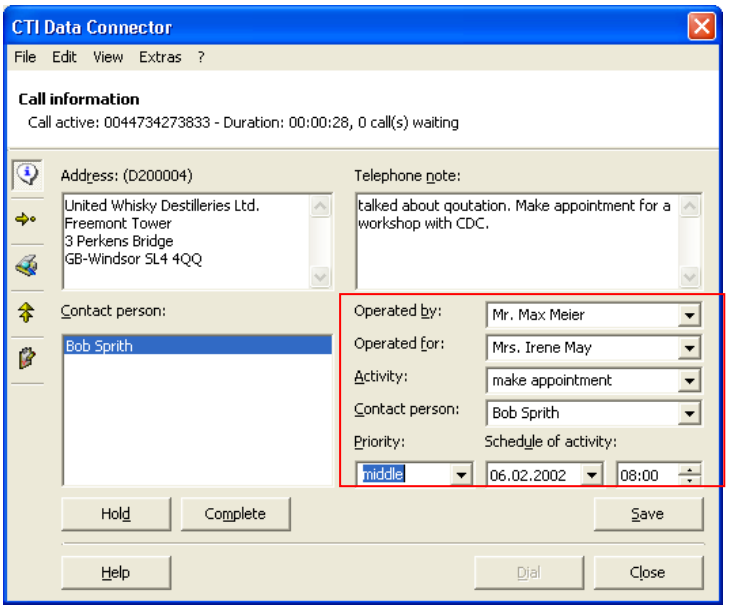
You have to activate this option in the node <profiles> for each database using <AddPhoneNote>1</AddPhoneNote>.

If this option is not configured, an image is displayed instead of the fields to type in a phone note.



5.1 Configuring the SQL Statements for Phone Note and Activity

The CDC client can make a direct access to your database to fill the list boxes for *Operated by*, *Operated for*, *Activity* and other fields with values from your database. This configuration is done in the CDC.xml file.

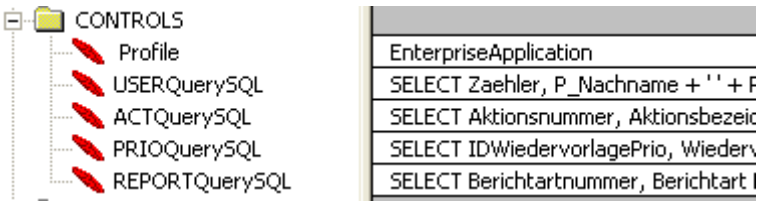


Definition for list box values

The definition of the list box values is done in the <CONTROLS> node. See chapter *Configuration File cdc.xml.*

At this time, only 1 section <CONTROLS> is supported.

Elements of the controls definition



Element name	Description
<Profile>	*Key of the profile definition* which is used for the data itself. **It must be identical to the element <key> in the node <profile>.**
<USERQuerySQL>	The SQL statement displays the user names (fields Operated by and Operated for). The first field is interpreted as the unique key, the second field is interpreted

as the display name. The list box can display all users of the Enterprise Application.

Example: SELECT user.IdxUser, user.Lastname + ' ' + user.Firstname AS username FROM user ORDER BY user.LastName, user.FirstName

<ACTQuerySQL> The SQL statement for forwarding activities.

The first field is interpreted as the unique key, the second field is interpreted as the display name. The activity combo box stores the last 10 entries if no activities statement is found.

Example: SELECT activities.IdxActivitiy, activities.ActivityName ORDER BY activities.ActivityName

<PRIOQuerySQL> The SQL statement for forwarding a priority.

The first field is interpreted as the unique key, the second field is interpreted as the display name. The priority combo box stores the last 10 entries if no priorities statement is found.

Example: SELECT priorities.IdxPriority, priorities.PriorityName ORDER BY priorities.PriorityName

<REPORTQuerySQL> The SQL statement for returning the types of a telephone call - inbound call, outbound call. If you specify this query, the phone note can be saved with that additional information.

The first field is interpreted as the unique key, the second field is interpreted as the display name. *Only the first two records are supported*. The first record should hold the telephone note type for *inbound calls*, the second for *outbound calls*.

Example: SELECT reporttypes.IdxReporttype, reporttypes.ReportName WHERE reporttypes.IdxReporttype =12 or reporttypes.IdxReporttype =15 ORDER BY reporttypes.ReportName (index 12 = record holding the text information for inbound call, index = 15 record holding the text information for outbound call)

5.2 The file CDCJOBS.XML

The file is in the folder **…\documents and settings\username\application data\ application exe name.** How to retrieve the folder name see chapter: *File cdccalls.xml*.

If the **SAVE** or **CLOSE** Button is pressed, CDC saves the telephone note in the file **cdcjobs.xml** and starts the external programme (see chapter *File cdccalls.xml*.). CDCJOBS.XML holds for **any event** one record under the root node <*CDC*>.

It can be defined, whether the cdcjobs.xml is created **each time an event occurs** or just **only once**. This is defined in the cdc.xml file, node <main>, **SaveJobOnlyOnce**. Only once is the default handling unless your application explicitly wants to be informed about each event. Informing about each event is necessary if you want to store updates on a phone note (e.g. user types in note, clicks on save, then modifies the note and closes the screen).

It is in the duty of the called programme to **delete the record if it is of no further use**. The record is indicated by its unique job identifier as main node under which all related call and job information are collected. The job information is the result of any user data entered or selected in the CDC front end. The elements are left blank if the no data is entered by the user in the corresponding front end control.

CDC calls the external programme with the command line parameter ***-sJOB_x***, where x is a 48 bytes string, indicating the unique job identifier of this job. The parameter followed by the ***–s*** is also the main node name of the record. The record contains all the information of the related call information record (except the sub nodes <*Choices*> and <*Contacts*>) and additionally the following elements:

Additional elements of the cdcjobs.xml

Element name	**Description**
<*StartTime*>	The regular starting time of the call
<*EndTime*>	The regular ending time of the call
<*Activated*>	-1 indicates that the call was activated (both parties took the call), 0 indicates that the call wasn't activated.
<*Completed*>	-1 indicates that the call was completed (one party terminated the call), 0 indicates that the call wasn't completed yet.
<*ManKey*>	The unique key of the company client, if the profile supports company clients. This is the first field indicated by the *<MANQuerySQL>* element in the profile definition.
<*ManText*>	The company client name, if the profile supports company clients. This is the second field indicated by the *<MANQuerySQL>* element in the profile definition.
<*Note*>	The users telephone note
<*OperatedByKey*>	The unique key of the user who operated the call. This is the first field indicated by the *<USERQuerySQL>* element in the controls definition.
<*OperatedByText*>	The display name of the user who operated the call. This is the second field indicated by the *<USERQuerySQL>* element in the controls definition.
<*OperatedForKey*>	The unique key of the user for whom the call was operated. This is the first field indicated by the *<USERQuerySQL>* element in the controls definition.
<*OperatedForText*>	The display name of the user for whom the call was operated. This is the second field indicated by the *<USERQuerySQL>* element in the controls definition.
<*ReportKey*>	The unique key of the telephone note type. This is the first field indicated by the *<REPORTQuerySQL>* element in the controls definition.

<ReportText>	The display name of the telephone note type. This is the second field indicated by the *<REPORTQuerySQL>* element in the controls definition.
<ActionKey>	The unique key of the further activity. This is the first field indicated by the *<ACTQuerySQL>* element in the controls definition.
<ActionText>	The display name of the further activity. This is the second field indicated by the *<ACTQuerySQL>* element in the controls definition.
<ActionForKey>	The unique key of the address' contact person with whom the further activity is planed.
<ActionForText>	The display name of the address' contact person with whom the further activity is planed.
<PriorityKey>	The unique key of the activity priority. This is the first field indicated by the *<PRIOQuerySQL>* element in the controls definition.
<PriorityText>	The display name of the activity priority. This is the first field indicated by the *<PRIOQuerySQL>* element in the controls definition.
<ActionDate>	The date on which the further activity is planed.
<ActionTime>	The time on which the further activity is planed.

Note

The date and time format within CTI Data Connector is set according the settings in Control Panel / Regional and Language Options.

If you use the file cdcjobs.xml you have to be aware, that the date and time format is set there also according to these settings. Your application which reads this information should consider that and use the same system settings.

6 Implementation for Browser based software

CDC can easily be used with a Browser based or hosted software.

6.1 Implement outgoing calls for Browser based software

To implement outgoing calls there are a lot of options which are explained in detail in the main chapter *Implement Outgoing Calls*

- Dial via Hyperlink – recommended option
- Dial via right mouse click within the Browser (only with internet explorer) – no programming necessary
- Dial via Hotkey – no programming necessary
- Invoke Dial.exe from the Browser
- Write the phone number into an XML file on – Dial via file

6.2 Implement incoming calls for Browser based software

Normally the database is hosted somewhere and accessed via the internet. To provide a fast caller identification, which is working even offline, the following steps are necessary:

- Download address information and phone numbers form the hosted database to a local database directly before the phone numbers are formatted
- Provide a login screen for the Browser application during the start of CDC
- Pop-up application to display caller data

6.2.1 Download Address Information

The address information has to be cached locally in a centralised database (normally once a day). Before CDC does the job to format the phone number, a programme can be invoked which downloads the data from the Web. See chapter *file cdccalls.xml, node <Externals>* - use TYPE = 4.

The complete URL of an account or contact could be stored in the database and used from the pop-up.exe (programme which is started after the caller identification is done) to open a new browser window with the caller data.

We could provide a sample access database.

6.2.2 Provide a login screen

To retrieve the login data for your Browser application, you could provide a login screen during the programme start of CDC See chapter *file cdccalls.xml, node <Externals>* - use TYPE = 5.

Store the login data in an encrypted format within the registry. The programme which downloads the data and the programme, which pop-ups the data in the Browser could retrieve this value.

6.2.3 Pop-Up application to display caller data

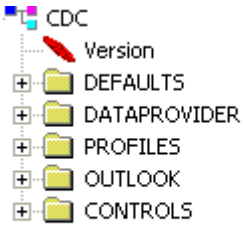
After the caller is identified, the predefined executable is invoked from the CDC application. The logic for this application could be as follows:

- Read caller data from cdccalls.xml file and delete the caller data within cdccalls.xml file
- Retrieve the record within the access database (record number is passed within the cdccalls.xml file)
- Read the URL from the record
- Retrieve the login data from the registry
- Start a new Browser with the URL data

7 Configuration file CDC.XML

You can pre-configure CDC with this file. It is stored on the server and should hold all global definitions.

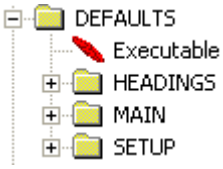
The file holds five main nodal points (nodes) under the root node *<CDC>*.



Some configuration settings are only available with the **OEM integration SDK and are not described here**. You may only modify the settings described in this chapter.

7.1 Main node <DEFAULTS>

Under this node all set up related configuration issues are collected. This node holds three further sub nodes named *<HEADINGS>*, *<MAIN>* and *<SETUP>* and a default entry for an external file (Executable)



Name	Description
Executable	The name of a windows executable programme of an application which is invoked for **all databases** (including MS-Outlook) **after** the file cdccalls.xml is written. If a specific Executable is defined for a database or MS-Outlook, then that Executable is used. Details see chapter *Invoke/Inform your application about a call.* A path to the file can be specified (f:\programms\myprog\pop-up.exe). If there is no path specified, the programme must be located within the CDC directory.

7.1.1 Sub node <Headings>

Do not modify these settings. This information will not be supported in the future.

7.1.2 Sub node <Main>

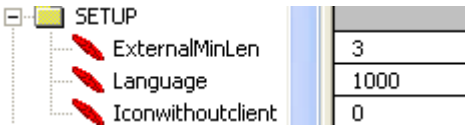
Most of these settings are only available in the OEM version. These settings can not be modified by a user.

Name	Description
SaveCallOnlyOnce	Defines how often call events like incoming call, call on hold or call terminated are signalled via the cdccalls.xml file. Only once or each event. Only once is the default handling unless your application explicitly wants to be informed about each event. 1 = yes, 0 = no. Details see chapter Event Handling and *The file CDCCALLS.XML*
SaveJobOnlyOnce	A Phone note or Activity can be signalled only once via the cdcjobs.xml file. 1 = yes, 0 = no. If you want to store updates on a phone note this switch should be set to 0. Details see chapter *Store a Phone Note and Activity*

7.1.3 Sub node <*SETUP*>

Here you can define **default values for the setup with the Wizard**. If not specified, build in default values are used. All values are written with the **same name to the registry**. The user can modify these settings within the wizard.

Like under the sub node <*MAIN*>, this node may contain any by CDC supported windows registry item with its name and value. The only difference is that the **user is able to change** those pre configuration defaults during runtime of the wizard.



In this example, the minimum length of phone numbers for an outgoing call is set to 3 and the language of the programme is set to German.

Name	Description
AlwaysOnTop	The CDC main window is always placed on top 1 = yes, 0 = no

Name	Description
DialDirectory	Directory where the file is placed when the function DialViaFile is used
DialViaFile	Dial with a filename (see chapter use a file for dialling) 0 = No, 1 = Yes
DontShowInternal	Do not show the internal calls 0 = No, 1 = Yes
ExternalDigit	Digit(s) to dial for establishing an external line
ExternalMinLen	Minimum length of phone numbers indicating an external call
FadeInbound	Activate the Desktop Notification Window for inbound calls 0 = No, 1 = Yes
FadeOutbound	Activate the Desktop Notification Window for outbound calls 0 = No, 1 = Yes
FadeTime	Time in seconds to display the Desktop Notification window
FindLikeChecks	Sets the value caller identification – truncate number of digits Default =4, US/Canada should be set to 0 Range: 0-4
HotKey	Keycodes of the Hotkey. To get the value for preconfiguration, just set the desired hotkey with the configuration wizard and retrieve the value from the same registry setting
HotKeyEnabled	Enable using a hotkey for dialling 0 = No, 1 = Yes
HotKeyModifier	Keycodes for Hotkey Combination 0 = None 1 = MENU 2 = CTRL 3 = CTRL + MENU 4 = SHIFT 5 = MENU + SHIFT 6 = CTRL + SHIFT 7 = CTRL + MENU + SHIFT
IconWithOutClient	IconWithoutClient affects the behaviour auf the CDC front-end. Your can change this item in the registry **on the fly** (CDC immediately reacts on this setting) to let CDC act as a middleware or with it's own client interface. 0 = with right mouse click on the CDC Icon you can start the CDC interface (same as double click) – Option *Open CDC Data Connector* is available 1 = user can't start the CDC interface (options available: format numbers, configuration wizard, error protocol, License Viewer, quit) 2 = user can't start the CDC interface (options available: error protocol, License Viewer, quit)

Name	Description
	With setting 1 and setting 2 the Client is not activated for incoming and outgoing calls (the same result than setting ShowInbound and ShowOutbound to 0). **Option 1 or 2 is recommended if CDC is used as a middleware**. You can use this setting to restrict or grant access to the CDC interface. The registry item is read each time an event in CDC occurs and can be changed during CDC is executed. A valid license for the CTI Client is necessary to use this option.
Language	Language identifier, 1000 = German, 2000 = English
ListEntries	Number of list entries stored in the redial and calling list. Maximum is 500
LogTapi	Logs all TAPI events in Applicationname.log 0 = No, 1 = Yes Default should be 0. Used only for debugging
OpenOutlookContact	Opens the outlook contact folder if a contact, stored in MS-Outlook, is identified 0 = No, 1 = Yes
PresetIn	Preset(s) delivered by the TSPI driver in front of the regular phone number for external *inbound* calls. Option: Prefixes in front of external calls within the wizard
PresetMode	Cut = 1 or add in front = 2 the configured presets
PresetOut	Preset(s) delivered by the TSPI driver in front of the regular phone number for external *outbound* calls. Option: Prefixes in front of external calls within the wizard
Registerrequest	0 = standard Windows TAPI handling 1 = a TAPI event for a call request from **all programmes** are sent to CDC and CDC handles it. This is recommended
ShowConfirm	Indicates whether the confirmation dialog of the wizard will be displayed = 1 or not = 0.
ShowInbound	Signal an incoming call 0 = No, 1 = Yes
ShowInboundMode	Display incoming calls 1 = when ringing 2 = when answering the phone
ShowIntro	Indicates whether the first page of the wizard (language selection) will be displayed or not 0 = No, 1 = Yes
ShowOutbound	Signal an outgoing call 0 = No, 1 = Yes
ShowOutboundMode	Display incoming calls 1 = when ringing 2 = when answering the phone

Name	Description
Simulate	Instead of using the configured TAPI line, CDC is acting with the integrated simulator. Yes = 1 or not = 0.
SuppMultiHits	Suppress multiple hits due to identical phone numbers 0 = No, 1 = Yes
TapiLine	The numeric line identifier of the TSPI driver (not used anymore – use TapiLineName instead)
TapiLineName	The name of the TAPI line. Identifier of the TSPI driver used by CDC, beginning at version 1.2
WebDial	Enable Web Dial via Internet Explorer, right mouse click. 0 = No, 1 = Yes (a valid license information is necessary)
WindowsDialer	Show windows dialer in case of placing an outbound call. 0 = No, 1 = Yes
WizWasStarted	0 = the configuration was never used. Start the configuration wizard (only valid when using cdcconf.dat) 1 = the configuration wizard has run at least once
WizParam	Start parameter for configuration wizard if WizWasStarted=0. If omitted, all wizard steps are available • **-M** – display only 1 page to select the TAPI driver and basic settings • **-S** – run wizard in silent mode. No interface is visible but all default settings and the settings from the cdconf.dat are applied
XML Directory	Directory for the cdcjobs.xml and cdccalls.xml file. Default directory is user directory (when not specified). See chapter: *The file cdccalls.xml*

7.2 Main node <Profiles>

This node can hold n elements (one for each database). See chapter *Define SQL statements* for all settings.

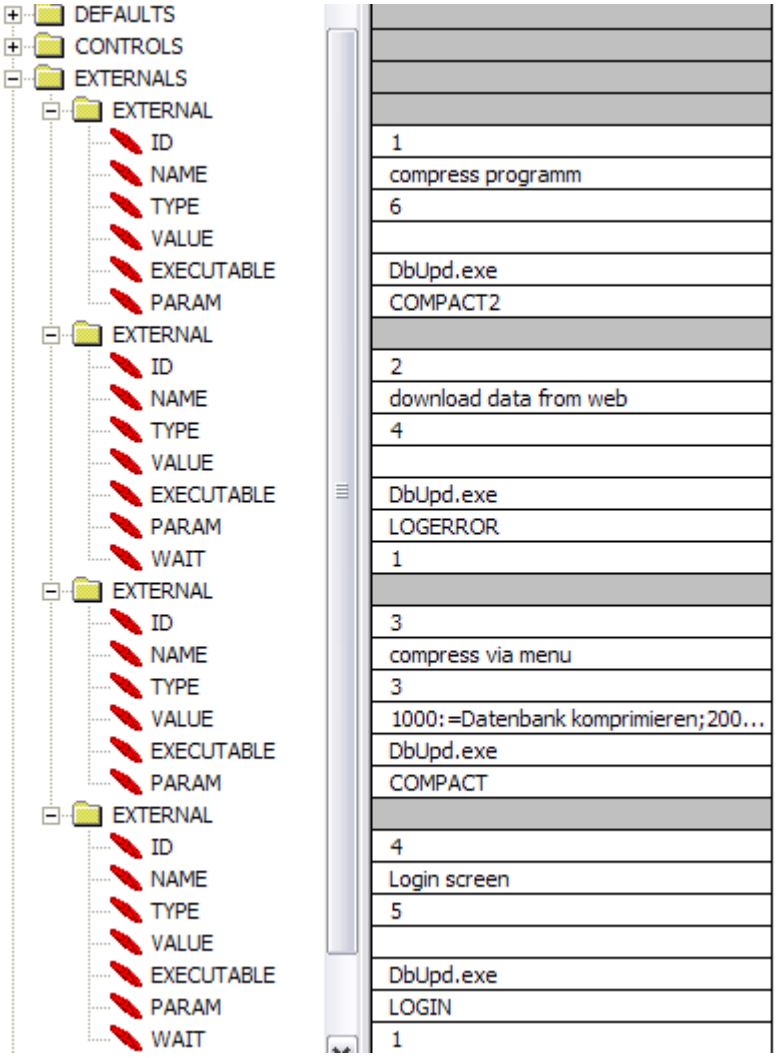
7.3 Main node <CONTROLS>

This node holds five elements, containing SQL statements for **the row sources definitions of the CDC controls (mainly list boxes)** and the key of the profile which is used for the data itself. See chapter *Configuring the SQL Statements for Phone Note and Activity* for all settings.

7.4 Main node <EXTERNALS>

You can define, if CDC should invoke other applications on different events. For details see chapter *Invoke External applications*

For each programme, which has to be started, a subnode <External> has to be created



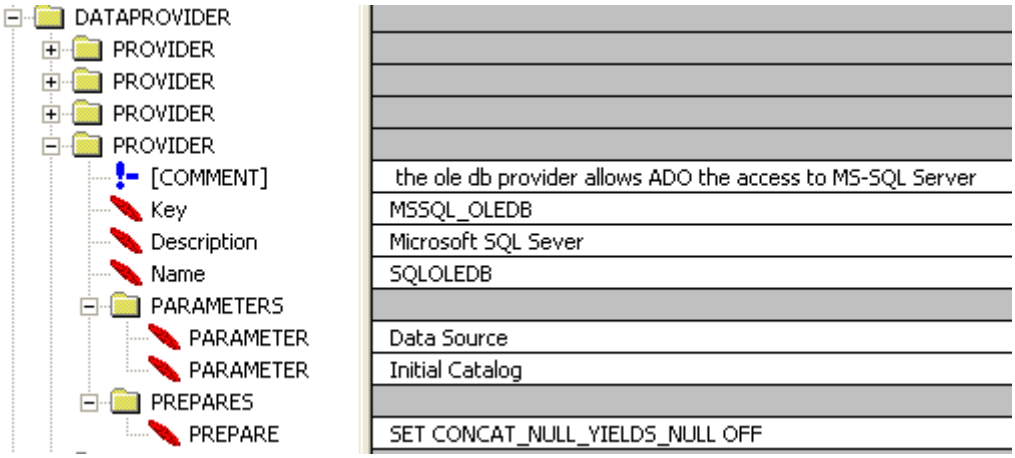
Name	**Description**
ID	Any numeric value – must be different for each node
Name	Short description of the functionality of the external programme like login screen, compress database
Type	Defines when the programme is invoked 1 = Interval (not applicable – do not use) 2 = Point in time (not applicable – do not use) 3 = New menu item within the CDC Client. Displayed within menu Extras – e.g. to run a completely additional application 4 = Before the formatting of the phone numbers is done – e.g. to copy data from Web to a local database 5 = During the programme start – e.g. to provide a login screen 6 = During the termination of the programme – e.g. to compress a database

Value	Depending on the setting in the field Type. Must have the same value like the Type field. 1 = interval in minutes 2 = time 3 = text for menu Extra in the CDC forntend If Type = 3 this is the definition for the text displayed within menu Extra for each language 1000:menu text in German; menu text in English <u>Example:</u> 1000:=Datenbank komprimieren;2000:=Compact database 4 = not applicable 5 = not applicable 6 = not applicable
Executable	Path and filename for the programme which should be invoked
PARAM	Parameters which should be passed to the executable
WAIT	Only valid if field TYPE is set to 4 or 5 Should CDC wait until the external programme has been terminated 0 = No, 1 = Yes

7.5 Main node <DATAPROVIDER>

This section holds 1 to n data provider definitions within XML elements under their separate sub nodes. The sub nodes' tag name must be *<PROVIDER>.* The following five data provider definitions and their keys are already included:

- Standard JET (Microsoft Access) data access (JET_OLEDB)
- ODBC data access by pre configured DSN (ODBC_DSN)
- ODBC data access by direct driver call (ODBC_DRIVER)
- MS-SQL Server data access (MSSQL_OLEDB)
- Oracle data access (ORACLE_OLEDB)



Elements of a data provider definition

Element name	Description
<Key>	Unique key for this data provider definition
<Description>	Data provider name displayed to the user
<Name>	ADO (Active Data Objects) provider name for the ADO connection

Under each data provider definition are two further sub nodes located.

7.5.1 Sub node <PARAMETERS>

Should contain one element for any parameter name, Required to establish a connection to the database.

7.5.2 Sub node <PREPARES>

May contain one element for any SQL Statement, which has to be sent to the database, before CDC sent its own SQL statement.

CDC comes with a predefined set of data providers. They can be configured as follows:

Microsoft Jet

Data Service Provider: Microsoft Jet
Data Source: Path and file name of the database e.g. c:\cdc\cdcdemodaten.cdc
You can specify a system database and a password

ODBC DSN (Data Source Name)

Data Service Provider: ODBC DSN
DSN: name of the ODBC Profile
You can specify a user and a password

ODBC Driver

Data Service Provider: ODBC Driver

You have to use the same parameters like you would use configuring an system DSN within windows

Direct Connection

You can make a direct connection for MS-SQL Server and Oracle to gain significant performance enhancements.

- **Microsoft**
  Data Service Provider: Microsoft
  Data Source: Name of the database Server
  Initial Catalog: Path and file name of the database e.g. c:\cdc\cdcdemodaten.mdf
- **Oracle**
  Data Source: Name of the database Server and Connection Information

If the field *user* and *password* is left empty an integrated login is executed.

8 Runtime modes of CDC

CDC supports 3 runtime modes. It can be executed as *single user system*, as a *client* and as a *server*. A single user is able to format the phone numbers of **all profile and local databases including MS-Outlook**. This is necessary, if CDC is used on a single machine or offline (e.g. on a notebook with a local database).

The client version of CDC is not able to prepare the phone numbers of profile databases (=enterprise databases). The client can prepare data from a local database or data from MS Outlook contact folders.

The server system does not support caller identification and the formatting of phone numbers of local databases and MS Outlook. It's only used to format and update the phone numbers of the main database(s).

The runtime mode of CDC is set during installation, depending on the installation options (single user system, multi user system, server setup).

Overview over the CDC versions

Functionality	**CDC Version**		
	Stand alone	Client	Server
data preparation profile database	X		X
data preparation local database	X	X	
data preparation MS Outlook	X	X	
caller identification	X	X	

9 Command line parameters of CDC

CDC supports command line parameters for starting CDC in server mode. Command line parameters overwrite any defined runtime mode.

Parameter	Description
-R	Starts CDC in server mode (run once mode), prepares the phone numbers and quit.
-Thh:mm	Starts CDC in server mode for preparing the phone number each time stated in the hh:mm parameter
-Mn	Starts CDC in server mode for preparing the phone number any n minutes.

We recommend using the task manager (Windows 2000, XP) to run CDC once every night. The big advantage of using the task manager is that the memory is freed after using CDC.

See online help and server setup (setupserver.exe) for more details.

10 Command line parameters of the configuration wizard

Command line parameters are added to the wizard for supporting administrators of wide networks. With these command line parameters the administrator is able to configure one client installation with a single command line. All parameters stated in the command line suppress the corresponding parameter in the configuration file CDC.XML.

Parameter	Description
-Ex	The numeric switch extension (direct dialling number) of the actual client installation, where x is the direct dialling number (do not use – no longer supported)
-Lx	The numeric line identifier of the TSPI driver used by CDC for the actual client installation, where x is the line identifier (do not use – no longer supported).
-S	Indicates that the wizard will not display any form or message box. The wizard is running in silent mode.
-M	Displays only 1 configuration page to select the TAPI driver and to configure basic settings

11 Using the ActiveX Interface

Please consult for the ActiveX interface the separate documentation ***Developer Documentation ActiveX interface.***